uc3m | Universidad Carlos III de Madrid

Grado en Ingeniería Electrónica y Automática

2018-2019

Trabajo Fin de Grado

ESCÁNER LECTOR DE COLOR PARA SUPERFICIES

Daniel Benito Bricio

Tutora

Nuria López Ruiz

02/07/2019

RESUMEN

Existen infinidad de personas cuyos trabajos dependen directamente de la utilización de colores. En la mayoría de los casos es complicado diferenciar los matices entre distintos colores ya que, en general, estos dependen tanto de factores externos, como la luz que incide sobre ellos en ese momento, como de las propias personas, dotando a la visión del color de cierta subjetividad. Además, hay usuarios que padecen problemas de visión, los cuales les generan dificultades en su día a día debido a la imposibilidad de diferenciar correctamente entre colores. Hoy en día existen sistemas capaces de reconocer los colores de diferentes objetos de forma muy precisa, aunque suelen tener un coste elevado y su uso está principalmente enfocado a la pintura y el arte.

El objetivo del estudio es crear un sistema basado en tecnologías de reconocimiento de color. El dispositivo diseñado tiene que ser capaz de ofrecer datos específicos de las características de los colores a las personas que busquen referencias de estos mismos. Con esta finalidad en mente, cabría preguntarse de qué manera se puede desarrollar un sistema, encargado del reconocimiento de colores, con alta precisión, que resulte al alcance de la mayor parte de los usuarios.

El sistema desarrollado se presenta en una estructura portátil, perfecta para su uso en cualquier situación sin que su transporte resulte un estorbo. Se ha utilizado un procesador Arduino para su desarrollo, ya que permite una amplia gama de posibilidades con buenos resultados a muy bajo coste. Además, este sistema se comunica, inalámbricamente, con una aplicación móvil. Esto es una gran ventaja para los usuarios que deseen utilizarlo ya que, hoy en día, la mayoría de las personas poseen un teléfono móvil o pueden adquirirlos a un precio reducido.

Tras las pruebas finales, realizadas con el sistema completo, los resultados obtenidos muestran una alta precisión del sistema. Ante una repetición de adquisiciones al mismo color, los resultados tienen un error despreciable. Las comparaciones se ejecutan correctamente, ofreciendo resultados muy similares al color medido. Estos resultados facilitan la necesidad de ciertas personas a conocer las características exactas del color que tienen delante, ofreciendo también la posibilidad de obtener los datos del número de referencia de un fabricante que distribuya ese color para una aplicación determinada, por ejemplo, pintura.

El sistema puede ayudar en muchos ámbitos a cualquiera de las personas descritas al comienzo. Además, se pueden realizar actualizaciones al sistema que le añadan nuevas características como, por ejemplo, una respuesta de sonido tras las comparaciones de colores o pintar la pantalla del color aproximado que se encuentra en la base de datos.

ABSTRACT

There are countless people whose jobs depend directly on the use of colours. Most of the times it is difficult to differentiate the nuances between different colours, considerate that, in general, they depend on external factors, such as how they are lighted up at that moment. Besides the large number of people who suffer from visual problems, which cause them difficulties in their day to day activities due to the impossibility to distinguish between colours. Nowadays there are systems which are capable of recognize the colours of different objects in a very precise way, although they usually have a high cost and their use is mainly focused on painting and art areas.

The objective of this study is to develop a system based on colour recognition technologies. The designed device has to be able to offer specific data of colour's characteristics to the people who look for references of them. With this purpose in mind, one might wonder how a system can be developed, responsible for the recognition of colours, with high precision, which is available to the majority of users.

The developed system is presented in a portable structure, perfect for his use in any situation without its transportation being a hindrance. An Arduino processor has been used for its development, since it allows the system to reach a wide range of possibilities with good results at a very low cost. In addition, this system communicates, wirelessly, with a mobile application. This is a great advantage for users who want to use it because, nowadays, most of people own a mobile phone or can buy it at a reduced price.

After the final tests, carried out with the complete system, the results obtained show a high precision of the system. Faced with a repetition of acquisitions of the same colour, the results have a negligible error. The comparisons are executed correctly, offering results very similar to the measured colour. These results facilitate the need for certain people to know the exact characteristics of the colour in front of them, also offering the possibility of obtaining the data of the reference number of a manufacturer that distributes that colour for a given application, for example, painting.

After the final tests, done with the system complete, the obtained results show a high precision of the system. compared with repeated acquisitions of the same colour, the results have a negligible error. The comparisons are executed correctly, offering results very similar to the measured colour. These results facilitate the need OF certain people to know the exact characteristics of the colour in front of them, also offering the possibility of obtaining the data of the manufacture's reference number that distributes that colour for a certain application, for example, painting.

Taking into account these results, the system can help many people in a lot of areas, which are described at the beginning. In addition, you can make updates to the system

to add new features, for example, a sound response after comparisons of colours or the possibility of painting the screen with the recognised colour.

DEDICATORIA

Quisiera agradecer, a las personas que más me han apoyado y más cerca han estado, la realización de este TFG.

A mi familia, que siempre han estado conmigo, apoyándome y animándome, cuando lo necesitaba. A mi compañera, encargada de corregir una y otra vez mis torpezas de expresión, siempre con una sonrisa y palabras de ánimo. A todos aquellos que han soportado mis repetitivas charlas acerca de ideas que iban y venían para el proyecto.

A todos, gracias por estar ahí, y espero, que esto solo sea un recuerdo más en el camino que nos queda.

Gracias.

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

1. INTRODUCCIÓN

1.1. Motivación del proyecto

Todo proceso de investigación es llevado a cabo gracias a una motivación previa, que lanza al investigador al estudio y desarrollo de nuevas tecnologías. A continuación, se exponen los motivos que han dado lugar a la investigación realizada y documentada en este informe:

- Primero, se pretende explicar las razones desde un punto de vista general y sociológico. A diario se desarrollan tecnologías que ofrecen comodidad a las personas y aumentan su calidad de vida. Estas nuevas tecnologías se acomodan a las necesidades de cada individuo, permitiendo alcanzar unas metas antes imposibles, así como abriéndole caminos nuevos, los cuales resultarían imposibles de cruzar de no ser por estas investigaciones.
  Poder formar parte del avance de estas tecnologías y otorgar mejoras a la calidad de vida de ciertos grupos de personas, resulta una motivación muy satisfactoria a la hora de comenzar un proceso de investigación.

- En segundo lugar, también existe una motivación más exhaustiva desde un punto de vista técnico acerca de una investigación detallada. Tras la realización del grado en ingeniería, existen muchos conceptos que solo se alcanzan a rozar. Investigaciones como esta permiten completar estos conocimientos acerca de instrumentos y conceptos antes desconocidos y útiles para un posible futuro. La elección de un proceso de investigación puede ser motivada por el deseo de poner a prueba lo aprendido, y obtener resultados de dichos conocimientos. Por otro lado, además se puede obtener motivación de investigaciones que permitan adquirir conocimientos nuevos, como es este caso, y que nos preparen mejor para el futuro.

En concreto, la motivación para realizar este proyecto consiste en desarrollar un sistema que sea capaz de reconocer los colores físicos, midiendo las características que los definen. Con ello se pretende eliminar la subjetividad que existe en la estimación del color percibido por el ojo humano, ya que este se ve afectado por diversas condiciones, ya sean de la propia persona (daltonismo, tricromacia) o del ambiente (luz exterior, contraste), imposibilitando la definición correcta el color.

A su vez, se pretende poner en marcha este sistema utilizando un Arduino conectado a un sensor de color con alta sensibilidad y encapsulado, garantizando así mantener las condiciones ambientales (iluminación) en cada medida, y obteniendo la medida del color físico con sus valores RGB (rojo, verde y azul).

1.2. Objetivos

El principal objetivo que a conseguir con este proyecto es, construir una base de datos que almacene las características de una gran variedad de colores, de diferentes fabricantes de pinturas, para que, cuando se realicen medidas de diferentes objetos, estas sean comparadas con las medidas guardadas de la base de datos y se obtenga una referencia, del color fabricado, más parecido al medido.

A continuación, se exponen otros objetivos marcados para este proyecto:

- Diseñar un software capaz de analizar diferentes colores, reconocerlos y almacenarlos. Debe poder distinguir colores con características similares, así como el tono o su saturación, permitiendo tener una alta sensibilidad en el proceso de reconocimiento.
- Debe tener una interfaz sencilla y clara, permitiendo su uso al mayor número de usuarios posibles, incluidas las personas no familiarizadas con el manejo diario de tecnologías. Para ello se pretende hacer posible la implementación del software en herramientas de uso diario como, por ejemplo, un teléfono móvil.
- Realizar una investigación acerca de los colores, sus características ópticas y de qué forma pueden ser percibidos utilizando un sensor electrónico en el rango del espectro visible.
- Diseñar un Hardware destinado al procesado de los datos, capaz de comunicarse con la herramienta encargada de su almacenamiento (teléfono móvil). Debe tener un tamaño que permita su manejo y transporte de forma sencilla. Diseñar una caja del menor tamaño posible donde almacenar el hardware del sistema con salidas para poder conectarlo al PC y adquirir los colores.

La consecución de los objetivos mencionados anteriormente permite adquirir competencias específicas no desarrolladas durante la especialización cursada en la universidad.

1.3. Impacto-socio económico

La investigación realizada está dedicada a mejorar algunos ámbitos sociales. Los entornos que más se benefician de los resultados obtenidos son los relacionados con el mundo de la pintura, así como los que en su día a día trabajan en el diseño, donde su principal herramienta es el color.

Empresas dedicadas a la fabricación y venta de pinturas pueden obtener beneficios con el uso del software, introduciendo en él datos de sus productos para que sirvan de referencia a otros usuarios. Con estos medios pueden darse a conocer al público y, además,

ofrecer directamente sus productos como solución al problema del usuario que utiliza el software.

También ofrece beneficios a usuarios cuyo oficio está directamente relacionado con los colores: pintores, diseñadores, etc... El ojo humano, dependiendo de la luminosidad del ambiente y la situación en la que se encuentre, no es capaz, en ocasiones, de diferenciar distintos tonos de un mismo color. El software permite realizar esta tarea, reconociendo el tono que se está evaluando y ofreciendo la posibilidad de adquirirlo mostrando los datos del fabricante y su número de referencia.

A aquellas personas con problemas para diferenciar colores, como aquellos que sufren de daltonismo, les abre un camino en el ámbito laboral permitiéndoles solucionar sus limitaciones en el reconocimiento de colores. Usando comandos de voz la aplicación puede ser adaptada para el uso de gente con dificultad o pérdida visual, ampliando sus oportunidades. Un ejemplo de uso para este caso es detectar cuándo un alimento no se encuentra en buen estado, si una manzana tiene un color distinto a su color natural, puede indicar que se encuentra en mal estado.

1.4. Estructura del documento

Como se ha documentado al comienzo del informe, el objetivo del proyecto se basa en el reconocimiento de colores y almacenamiento de sus datos. Esta investigación ha pasado por varias fases a lo largo del proyecto. Para expresarlo de la manera más clara posible y resulte comprensible, a continuación, se detalla la estructura del documento:

- En el primer apartado se expone la base teórica de la investigación. En esta parte se explica el funcionamiento del sensor y la parte de la física óptica centrada en el rango visible del espectro, donde se encuentran los colores detectables por el sensor.
- La siguiente sección del documento está dividida en dos partes, el desarrollo del software y el hardware. Se ha explicado de forma separada para asegurar la comprensión completa del sistema y, a su vez, poder hacer el suficiente hincapié en cada parte.
- Tras los apartados anteriores se encuentran las pruebas finales. Una vez el sistema está terminado se realizan las primeras pruebas registrando una serie de colores y posteriormente realizando comparaciones en distintas situaciones.
- La siguiente parte del informe se corresponde con el presupuesto y la planificación del proyecto. Se hace un estudio de los gastos requeridos para la elaboración del proyecto y una planificación del tiempo empleado en él.
- Finalmente, se encuentran las conclusiones finales al desarrollo de este proyecto completo. Se hace un repaso de los objetivos que habían sido propuestos inicial-

mente y si han sido conseguidos al final. También, se elaboran una serie de trabajos futuros los cuales pueden mejorar el sistema desarrollado.

2. ESTADO DEL ARTE

2.1. Introducción

En un proyecto dedicado a la medida y reconocimiento de los colores, resulta imprescindible aclarar los conceptos teóricos que rodean al color, ya que estos han sido utilizados en el proyecto. En este apartado se comienza explicando en detalle el concepto de ”Colorimetría”. Se hace un repaso acerca de los conceptos físicos que componen la teoría del color por ejemplo, las características físicas que poseen los colores o cómo estos son emitidos por los objetos que nos rodean. También se hablará acerca de la posibilidad de percibir mediante el uso de la electrónica los diferentes colores utilizando sensores especializados.

Se hace hincapié en los sistemas encargados de la detección de los colores explicando su funcionamiento de manera sencilla y dando ejemplos de campos donde puedan ser utilizadas estas herramientas.

Por otra parte, se dedica este apartado a la realización de un estudio comparativo acerca de productos idénticos o parecidos al desarrollado en este proyecto. Se necesita obtener información acerca de las ofertas de equipos comerciales si se pretende que el sistema desarrollado tenga éxito comercial.

2.2. Colorimetría y lectura del color

La colorimetría se encarga de estudiar y cuantificar los colores, clasificándolos y diferenciándolos mediante sus características principales que los definen [1]:

- **Luminosidad:** La luminosidad de un color está definida por la cantidad de energía que llega a una zona por unidad de tiempo. Un color que posea un nivel de luminosidad muy bajo (cerca del 0 %) se verá negro.
- **Saturación:** La saturación corresponde con la pureza del color. Ante un nivel de saturación alto, el color se ve de forma muy pura, mientras que, si la saturación es baja, el color pasa a tener un tono gris. La mezcla entre una luminosidad del 100 % y una saturación del 0 % constituyen el color blanco.
- **Tono/Matiz:** El tono corresponde con la longitud de onda que tiene por preferencia el color que se está midiendo. Abarca todo el espectro visible desde el rojo al violeta.

Combinando estas características se forma el total de la escala de colores que pueden percibirse. Los colores formados se identifican utilizando unos métodos denominados

”Modelos de color” [2]. Se encuentran cuatro modelos que se explican a continuación:

- Hex: Mediante la numeración hexadecimal se pueden definir los diferentes colores usando un código de seis dígitos con la siguiente estructura, RRGGBB, correspondiendo a los valores, en sistema hexadecimal, de los colores verde, azul y rojo que forman el color medido tomando en cuenta su luminosidad y saturación también.
- HSV: Se forma mediante tres dígitos correspondiendo cada uno de ellos a: Matiz con valores entre 0 y 359 (H), saturación comprendido entre 0 % y 100 % (S) y valor también con valores entre 0 % y 100 % (V).
- HSL: Método muy parecido al anterior, se forma mediante tres dígitos correspondiendo cada uno de ellos a: Matiz con valores entre 0 y 359 (H), Saturación comprendido entre 0 % y 100 % (S) y Luminosidad también con valores entre 0 % y 100 % (L). La principal diferencia entre estos dos modelos, HSV y HSL, es que en HSV el valor de la saturación varía entre el color puro (100 %) al blanco (0 %) y el Valor varía entre el color (100 %) y el negro (0 %), mientras que en HSL la saturación varía entre el color puro (100 %) al gris (0 %) y la luminosidad varía entre el blanco (100 %) y el negro (0 %).
- RGB: Este es el método más utilizado de los cuatro. Consiste en un código de tres siglas RGB, cada una de ellas obtiene un valor comprendido entre 0 y 4096 (Valor máximo que otorga el sensor usado), estos valores corresponden con la cantidad de rojo (R), verde (G) y azul (B) que poseen los colores medidos, así como su saturación y luminosidad. Como en el proyecto se pretende establecer una base de datos muy genérica, este es el método que se utilizará para registrar las adquisiciones.

A continuación, se presenta un ejemplo donde se puede apreciar, para un mismo color, las diferentes maneras de nombrarlo utilizando los modelos explicados antes.

Fig. 2.1. Ejemplo Color.

Estos son los valores que presentan los diferentes modelos para el color de la figura 2.1:

- Hex: 954FB3.
- HSV: 282 - 56 % - 70 %.
- HSL: 282 - 40 % - 51 %.

- RGB: 149 - 79 - 179.

Variando uno a uno los valores de cada modelo, se puede formar el total de la escala de colores que pueden ser percibidos.

Con esto queda claro de qué forma pueden ser clasificados los diferentes colores y en qué están basadas las diferentes formas de clasificación. Es necesario profundizar más explicando los conceptos físicos del color para poder comprender mejor el funcionamiento de los diferentes métodos de reconocimiento.

Los colores se definen como estímulos psicofísicos producidos por la radiación que emiten los objetos [3]. El ojo humano posee células receptoras, conos y bastones, capaces de registrar la radiación electromagnética emitida y permitiendo la visión dentro del rango espectral visible (figura 2.2).

Fig. 2.2. Rango del espectro visible [4]

Los conos están formados por moléculas sensibles a las diferentes longitudes de onda de tres colores: verde, azul y rojo. La luz reflejada de un plátano estimula los conos rojos y verdes, estos envían la señal al cerebro, procesando la información, lo que resulta en la visualización del color amarillo [5].

La suma de las distintas longitudes de onda que forman el rango del espectro que es visible, da lugar a la denominada luz blanca. Cuando la luz blanca incide sobre la superficie de un objeto es reflejada, absorbida o transmitida por la superficie. La transmitancia permite pasar la luz a través de la superficie, de esta forma el objeto se ve transparente y no emite ningún color visible. Un ejemplo muy conocido es la transmitancia de luz blanca mediante un prisma, donde la longitud de onda de la luz sufre una refracción producida por el cambio de medio [3].

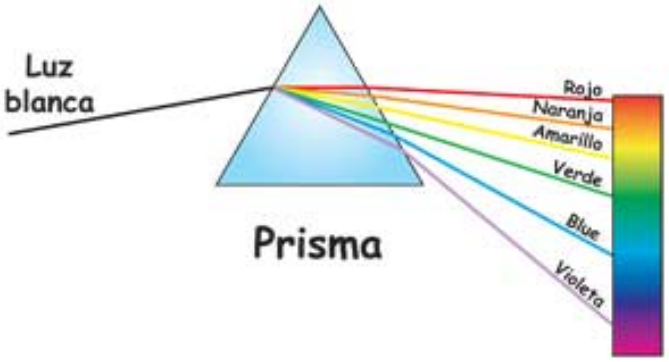


Fig. 2.3. Descomposición de la luz blanca en diferentes longitudes de onda del espectro visible dando lugar a los colores [6]

Las distintas longitudes de onda poseen un ángulo de refracción diferente provocando la separación de la luz blanca, como se ve en la figura 2.3, pudiendo observarse la composición de colores que la forman.

La reflectancia es la principal propiedad que hace posible visualizar los colores de los objetos. Cuando se produce una reflexión en la superficie del objeto donde incide la luz blanca, ésta es absorbida en su mayor parte, reflejando la radiación perteneciente a una longitud de onda concreta. Por ejemplo, una pelota roja, se ve de ese color porque el material del que está hecha la pelota, o el pigmento que la recubre, absorbe la energía recibida de la luz excepto la perteneciente al color rojo (700 nm), ésta será reflejada por la superficie del objeto siendo captada por el ojo humano. Los objetos también son capaces de emitir luz por sí mismos, esto ocurre cuando se encuentra a una temperatura superior de $500^{o}C$.

Se ha hablado anteriormente de la capacidad del ojo humano para detectar los colores, activando moléculas pertenecientes a tres colores específicos (rojo, verde y azul). Este grupo de colores es llamado ”colores primarios aditivos”, ya que combinándolos se pueden formar el resto de colores del espectro. La unión de estos colores por parejas forma los llamados ”colores sustractivos primarios” que son: magenta formado por rojo y azul, amarillo formado por rojo y verde y cian formado por verde y azul. La unión de los colores primarios forma la luz blanca y la unión de los colores sustractivos primarios forma el color negro, que representa la ausencia de color) [1].

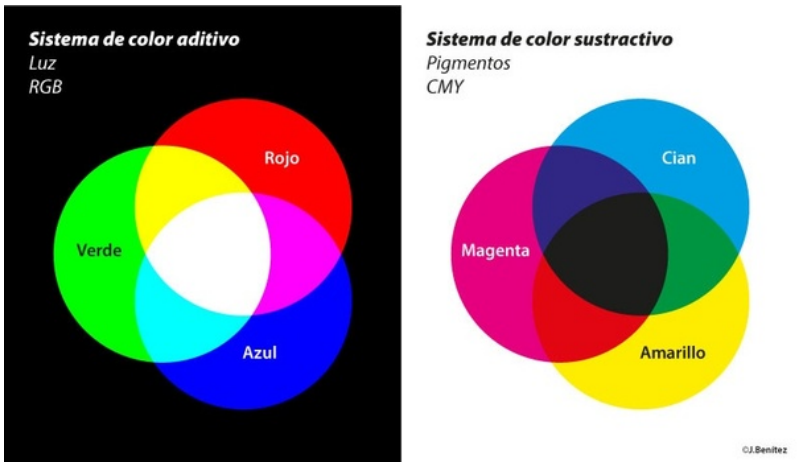


Fig. 2.4. Colores Primarios [7]

El sistema de color sustractivo es utilizado principalmente para la fabricación de pinturas o sistemas de impresión a color. Los artistas utilizarían este sistema para realizar sus cuadros y pinturas.

Actualmente el sistema de color aditivo es utilizado por todas las tecnologías que capturan o reproducen imágenes y dependen de la emisión de luz, ejemplos de estas tecnologías son: cámaras fotográficas, televisiones, monitores de ordenador, proyectores, detectores de color, etc...

Las tecnologías basadas en la reproducción de imágenes utilizan sistemas basados en el modelo RGB. Se trata de una matriz de píxeles iluminados cada uno por leds de distintas longitudes de onda (rojo, verde y azul). Como se puede observar en la figura 2.5, el color se produce por la combinación de los tres colores primarios en diferentes cantidades, permitiendo obtener cualquier color del espectro [8].

Fig. 2.5. Modelo RGB [9]

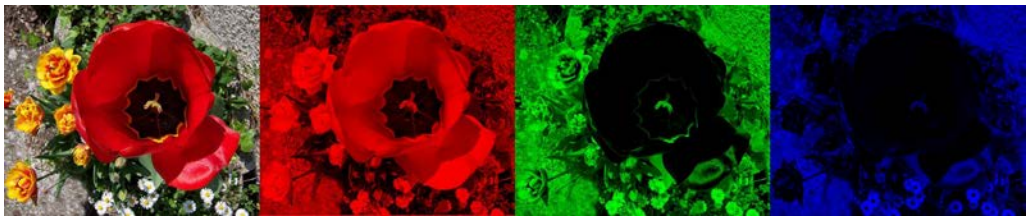


Fig. 2.6. Modelo RGB 2 [10]

En la Figura 2.6 pueden verse de qué forma se iluminan los leds mencionados anteriormente para formar la imagen de la izquierda. La flor, al tener un tono rojo prácticamente puro, solo es iluminada por los leds rojos y los leds verde y azul permanecen con una intensidad muy baja.

Por otro lado, se encuentran las tecnologías basadas en la detección de imágenes como, por ejemplo, las cámaras fotográficas. Estas tecnologías funcionan mediante el uso de detectores que recogen la radiación emitida por los objetos de nuestro alrededor [11]. La principal característica de los detectores es su rango espectral, existen detectores para todo el espectro electromagnético. Los detectores utilizados por las cámaras fotográficas tienen su rango espectral centrado en el espectro visible, están formados por una matriz de píxeles donde cada uno recibe la radiación emitida por los objetos a los que está enfocado. Actualmente, se encuentran dos tipos de tecnologías utilizadas en la fabricación de sensores de cámaras digitales: CMOS y CCD [12].

Además de los detectores utilizados en las cámaras fotográficas existen otro tipo de detectores, que permiten clasificar por color las imágenes medidas. Para esté trabajo se ha utilizado el sensor de color S9706 cuyo fabricante es Hamamatsu Photonics, compañía japonesa encargada de la fabricación de sensores ópticos [13].

Fig. 2.7. Sensor Color S9706 [13]

Este sensor de color posee una matriz de detectores de 9x9, donde 27 tienen su rango espectral centrado en la longitud de onda perteneciente al color verde (540 nm), 27 centrados en la longitud de onda perteneciente al color rojo (615 nm) y otros 27 en la longitud de onda perteneciente al color azul (465 nm). Con esta configuración el sensor detecta las radiaciones emitidas por los denominados ”colores primarios” como se puede apreciar en la figura 2.8, de esta forma se pueden detectar las diferentes cantidades de estos

tres colores que forman el color medido. Volviendo a la figura 2.6, puede ser interpretada en el sentido contrario, los diferentes detectores verían las tres imágenes de la derecha, obteniendo los niveles de los colores, rojo, verde y azul, que formarían la imagen de la izquierda.

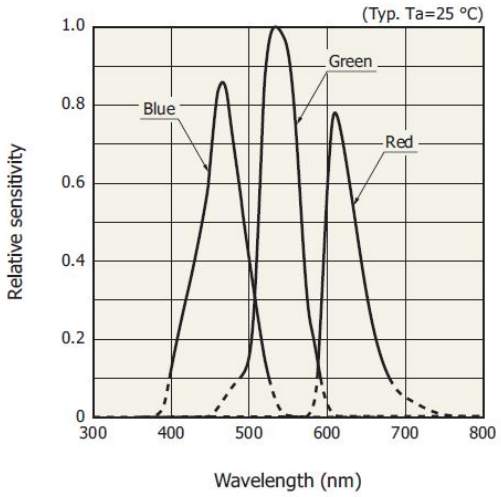


Fig. 2.8. Rango espectral de los detectores que componen el sensor [13].

El sensor ofrece una salida compuesta por 36 bits perteneciendo los 12 primeros bits a la medida correspondiente con los detectores enfocados en el rango espectral del color rojo, los siguientes 12 bits corresponderían con el color verde y los últimos con el azul. A partir de estos datos y tras convertirlos a un número decimal, se obtiene un valor comprendido entre 0 y 4095 para cada color, este modelo de sensor ofrece una sensibilidad lo suficientemente grande como para poder detectar una amplia gama de colores. Con estos valores se pueden identificar los diferentes tipos de colores medidos utilizando el modelo de color nombrado anteriormente como ”RGB”, permitiendo clasificar y reconocer diferentes colores.

Previo al uso del detector, es necesario establecer un tiempo de integración óptimo para el uso que se le quiera dar. El tiempo de integración en los detectores corresponde con un valor numérico medido en unidad de tiempo, este valor establece el tiempo que el sensor tarda en realizar una captura del color integrando la luz que llega durante ese periodo de tiempo [14]. El tiempo de integración (IT) define la sensibilidad de un detector ya que, el rango de medida es mayor o menor según lo sea este. Tener un IT alto puede provocar la saturación rápida del detector provocando que su nivel de medida sea el máximo, siempre a partir de una luminosidad media. En cambio, para un nivel de IT bajo, el detector puede no percibir los colores con luminosidad baja y obtener siempre el valor de offset que otorga el detector, se visualiza cuando no está midiendo nada (Negro) y es provocado por el ruido de los componentes electrónicos. Por estas causas es necesario procurar mantener siempre la misma luminosidad en cada medida, y escoger en base a esta un IT que se corresponda con las necesidades del proyecto.

2.3. Escalas de color comerciales

Una escala de colores puede definirse como la tabla que muestra las distintas formas en que se ven de los colores modificando sus atributos característicos (Tono, Luminosidad y Saturación) [15]. Se pueden encontrar dos tipos de escalas de color:

- Escala de color acromática (Figura 2.9): Es una escala de grises donde el color pasa del blanco al negro modificando su luminosidad. Se utiliza principalmente para escoger la luminosidad y saturación previa de los colores a utilizar, una vez hecha esta elección se escoge el color modificando solo su tono.

Fig. 2.9. Escala Acromática [16]

- Escala de color cromáticas (Figura 2.10): Es la escala más completa, conforma todos los colores posibles modificando sus tres características. Las diferentes tonalidades dan forma a toda la variación de colores existente, del morado al rojo, mientras, la saturación convierte estos colores de puros a más claros hasta llegar al gris.

Fig. 2.10. Escala cromática [16]

Con estas dos escalas de colores se pueden escoger los colores deseados para la utilidad que se les quiera dar. Con la escala acromática se escoge el nivel de luminosidad que debe tener ese color, una vez escogido este valor se pasa a una escala cromática donde los colores correspondientes a la anterior luminosidad pasan a variar dependiendo exclusivamente de su tono y saturación.

Existen escalas que representan las tres características al mismo tiempo (Figura 2.11). Estas escalas se conocen como gamas de color, y son menos utilizadas por tener una forma

tridimensional. Resulta más cómodo trabajar con tablas de dos dimensiones, ya que son más fáciles de representar y dibujar, además, otorgan gran facilidad a la hora de escoger un color específico.

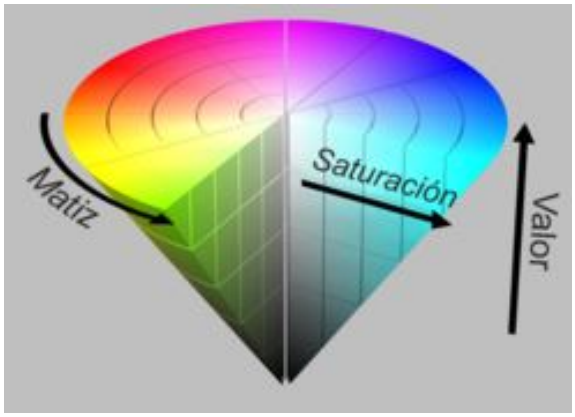


Fig. 2.11. Gama de color [17].

Para la realización del proyecto se han tenido en cuenta diversas escalas de colores comerciales. Cada fabricante de pinturas y diseñador expone sus propias escalas de colores y ha sido necesario estudiar varias de ellas para poder introducir los datos de sus respectivos colores en la aplicación con el fin de ser reconocidos cuando se miran colores de objetos cotidianos. Los principales fabricantes de colores, o marcas de pinturas, utilizados como referencia son: Titanlux, Bruguer o Pivema.

2.4. Sistemas actuales para reconocimiento de color

A continuación, se procede a realizar un desarrollo de los diferentes equipos comercializados que tienen relación directa con la tecnología estudiada en este proyecto. La finalidad es, comparar el funcionamiento de los equipos encontrados con el aquí desarrollado, y, obtener información acerca de las diferentes maneras por las que se consiguen los mismos resultados.

Los equipos estudiados se presentan con el nombre dado por sus fabricantes:

- Uno de los principales usos que se le ha dado a esta tecnología es, la capacidad de poder dibujar, utilizando un bolígrafo, con cualquier color que se encuentre en los alrededores. Estos equipos, están basados en el mismo principio estudiado para este proyecto, con el uso de sensores, detectan las características del color que se desea medir, y, utilizan los datos recogidos para generar combinaciones de tinta específicas y obtener un color similar al medido. Varios ejemplos de estos equipos son:

    1. **Scribble [18]:** Es un bolígrafo que posee un detector de color en la parte superior. Cuando el detector es posado sobre un objeto, este, detecta el color

de la superficie del objeto, consiguiendo los valores que lo forman. Con estos datos, internamente se ajustan los niveles de tinta de los colores primarios para generar el color medido y poder dibujar con él.

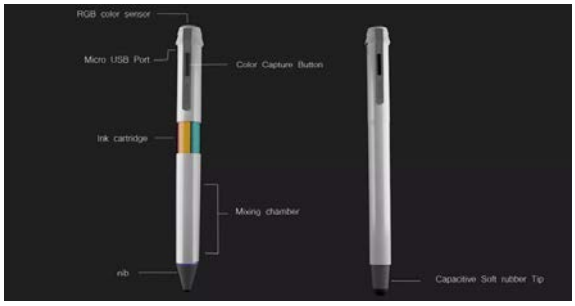


Fig. 2.12. Bolígrafo Scribble [18]

2. **Cronzy [19]:** Este bolígrafo es más parecido al diseño del proyecto. Al igual que el anterior, posee un sensor en la parte superior que se encarga de reconocer el color para su posterior procesado y generar la tinta adecuada. Como añadido, este aparato tiene la capacidad de conectarse, inalámbricamente, a una aplicación móvil, esta aplicación almacena los datos de los colores que el usuario ha medido, permitiendo reutilizar un color ya medido las veces que se desee sin necesidad de realizar una nueva adquisición.

Fig. 2.13. Bolígrafo Cronzy [19].

La principal diferencia observable entre estos equipos y el proyecto desarrollado es la diferencia de tamaño entre ellos. La decisión de utilizar un sistema Arduino, junto con el resto de componentes (FTDI y antena bluetooth) provoca un aumento en el tamaño final del sistema, pero facilita su manejo. Otra diferencia es la falta de una base de datos capaz de almacenar y comparar grandes cantidades de colores, en lugar de esto, están centrados en utilizar la medida realizada en el momento hasta que se realice una nueva. A parte de estas diferencias, la tecnología utilizada es la misma. La imposibilidad de uso de estos equipos impide que se pueda realizar una descripción más detallada exponiendo posibles mejoras y fallos que pudieran presentar.

- Se han encontrado equipos encargados del reconocimiento de colores orientado a las personas con problemas visuales. Uno de ellos es un equipo formado por una cámara, encargada de tomar adquisición de lo que el usuario está viendo. Esta herramienta transmite diferentes sonidos al usuario dependiendo del color del objeto que este viendo. Esta tecnología se aleja más de lo estudiado para este proyecto, pero la base sigue siendo la misma, reconocer colores.

  Una vez más, al no poder utilizar estos equipos resulta complicado establecer claras ventajas o desventajas entre ellos y el desarrollado.

2.5. Marco regulador

No se recogen leyes acerca de la implementación descrita en el trabajo por lo que, resulta imposible realizar un análisis acerca de esta legislación. El proyecto se compone de un sistema físico encargado de medir la luz que le llega y una aplicación instalada en el teléfono móvil de cada usuario. La aplicación no requiere de conexión directa a internet por lo que no existen riesgos con las privacidad y seguridad de los datos personales. La aplicación no requiere de la interacción entre usuarios, está pensada para el uso individual por lo que no supone riesgos éticos ni laborales.

No se recoge información acerca del software y hardware utilizado en la agencia digital de innovación pública (ADIP) acerca de los estándares técnicos relacionados con esta tecnología.

3. DISEÑO DEL EQUIPO

En el siguiente apartado se expondrá detalladamente el diseño del equipo completo desarrollado para el proyecto. Se realizará una descripción cronológica del desarrollo de las distintas partes que componen el sistema completo, así como una puntualización de los distintos obstáculos encontrados durante el proceso de creación y las diferentes maneras por las cuales estos han sido solventados.

Para la realización del sistema completo ha sido necesario el desarrollo en paralelo de las dos partes que lo componen. La parte del software (SW) está compuesta por los códigos utilizados en las distintas plataformas mediante las cuales los datos son estudiados y almacenados a partir de varios algoritmos especialmente desarrollados. El Hardware (HW) está formado por los componentes utilizados conectados entre sí para asegurar su funcionamiento y el encapsulado final del sistema diseñado para obtener los mejores resultados.

3.1. Hardware

Se comienza explicando el proceso llevado a cabo para la realización del hardware. Antes de comenzar a explicar el sistema, es necesario listar y comentar los componentes utilizados en el desarrollo del proyecto:

- Arduino Mini [20]: Es el cerebro del hardware, encargado de alimentar y coordinar el resto de componentes, todos ellos conectados a él. Manda las señales necesarias en el momento adecuado a cada componente para que estos interactúen entre ellos adquiriendo los datos, tratándolos y mandándolos.

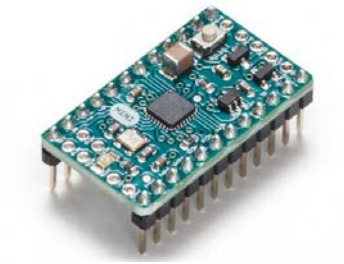


Fig. 3.1. Arduino Mini [20]

- Módulo bluetooth HC-05 [21]: Este componente se conecta directamente al Arduino, recibe los datos adquiridos después de haber sido tratados, y los envía a través de conexión bluetooth a cualquier aparato que esté conectado a este.

Fig. 3.2. Módulo Bluetooth [21].

- Placa FTDI [22]: Es un conversor de datos, se conecta directamente al Arduino permitiendo la conexión entre este y un PC. Esta conexión permite mandar al Arduino el software diseñado para dar las órdenes necesarias al sistema. También es la forma de alimentación del hardware durante el proceso de calibración y prueba, más adelante puede ser alimentado mediante una batería permitiendo su portabilidad.

Fig. 3.3. FTDI [22].

- Led Blanco [23]: Es el encargado de mantener constante la luminosidad en cada medida realizada con el sistema. El led se encarga de iluminar la muestra de color, sin que la luz incida directamente en el detector.

Fig. 3.4. Led Blanco [23].

- Sensor de color S9706 [13]: Es el corazón del sistema, encargado de las adquisiciones de los colores, envía la información de cada color al Arduino para ser tratado (Figura 2.7).

Estos componentes son los que forman el núcleo del Hardware, son las principales herramientas encargadas de la adquisición y el tratamiento posterior de los datos de los diferentes colores.

Durante el desarrollo del sistema hardware se utilizó una protoboard encargada de conectar los principales componentes entre ellos, esto resultó en un modelo principal de sistema utilizado para comenzar las primeras pruebas entre los diferentes componentes y los principales ajustes realizados para el buen funcionamiento de cada uno de ellos.

Las primeras pruebas se basaron en conseguir la conexión y comunicación entre los diferentes componentes. Dada la falta de experiencia trabajando con procesadores Arduino fue necesario el estudio y práctica con el mismo para lograr el correcto funcionamiento del sistema.

Una vez conseguida la comunicación entre los diferentes componentes el siguiente paso consiste en simplificar y mejorar el diseño de las conexiones entre ellos. La placa protoboard no permite el establecimiento del diseño en un encapsulado y la cantidad de cables entre los componentes resulta en un diseño muy tosco y frágil. Por ello se diseña una placa sobre la que los componentes serán soldados en ella simplificando el diseño y reforzando el sistema.

Para la realización de la placa se ha utilizado la aplicación ”Kicad” [24] encargada del diseño de circuitos y placas. Una vez más, la ausencia de experiencia en el uso de esta aplicación resultó en el estudio y pruebas en torno a él, utilizando los manuales de uso encontrados [25], obteniendo finalmente la placa con el circuito del sistema. En esta, para la simplificar el diseño se utilizó, para las conexiones entre tierras, el diseño de una superficie de cobre alrededor de toda la placa, conectando el resto de componentes directamente entre ellos con hilos del mismo material. Esto puede verse en la figura 3.5.

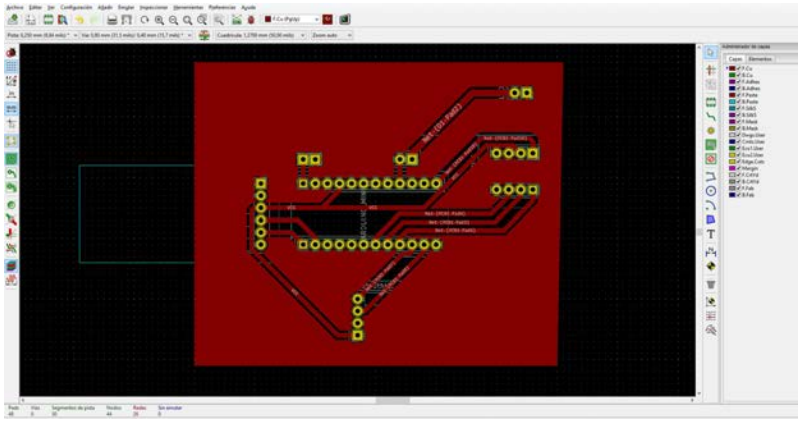


Fig. 3.5. Placa diseñada con la aplicación Kicad.

Una vez finalizado el diseño de la placa se procedió a la impresión de esta en un modelo físico.

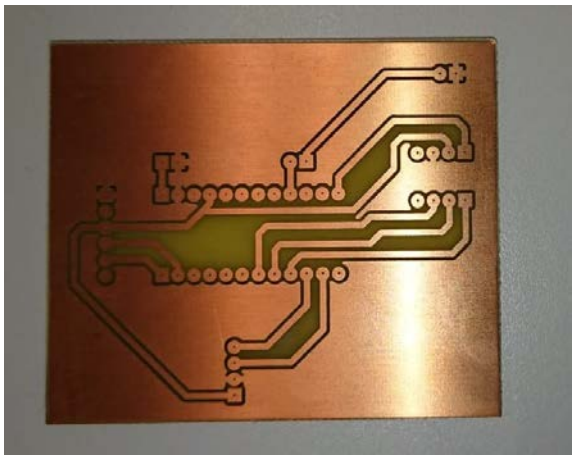


Fig. 3.6. Placa taladrada.

Tras realizar los taladros necesarios se procedió al soldaje de los componentes. Para mayor seguridad, los componentes no fueron soldados directamente en la placa. En su lugar, fueron soldados tiras de pines donde los componentes son enganchados posteriormente, como puede observarse en la figura 3.7

Esto permite, en caso de ser necesario, reemplazar un componente, quizá estropeado o anticuado, por uno nuevo, permitiendo mejorar si se quiere el sistema y ampliando sus capacidades.

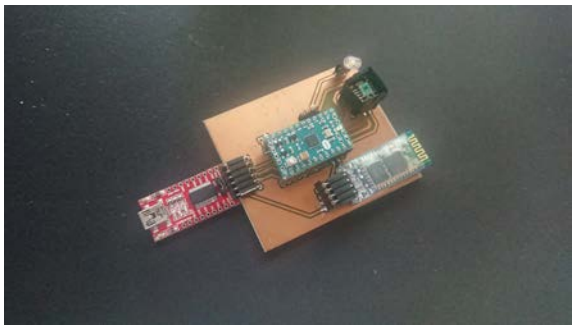


Fig. 3.7. Circuito integrado en la placa soldada.

Una vez se tiene el sistema totalmente integrado en la placa y tras realizar las pruebas necesarias que confirman el buen funcionamiento y la buena conexión existente entre todos los componentes, se procede al encapsulado del sistema. Mediante una caja de encapsulado, de color negro y de material plástico, se habilita para conseguir introducir el sistema en su interior, permitiendo la posibilidad de medir los colores realizando un agujero en la parte superior de ésta, situado justo encima del detector y el led, permitiendo la iluminación del color medido y su adquisición.

También se realiza un segundo agujero situado en un lateral de la caja, permitiendo la salida del conector encargado de asegurar la conexión entre el Arduino y el PC. Esto garantiza la conexión del sistema y la posible alimentación de este sin necesidad de liberarlo del encapsulado.

La caja utilizada ajusta la tapa a su base mediante cuatro tornillos, cerrando el sistema con seguridad y, a su vez, ofreciendo la posibilidad de liberarlo en caso de que fuese necesario el cambio de un componente o la mejora del sistema.

Fig. 3.8. Imagen del sistema encapsulado.

Con esto el sistema está completo y en pleno funcionamiento. Ya permite las distintas adquisiciones de colores totalmente aisladas de la luz exterior.

3.1.1. Consideraciones del hardware

En este apartado se detallarán las distintas consideraciones tenidas en cuenta a la hora de la realización del hardware del sistema. Se expondrán cronológicamente, según fueron surgiendo a lo largo del proceso de diseño del hardware.

Estas consideraciones serán expuestas en base a los cambios realizados respecto al diseño inicial, habiéndose comprobado de esta forma su mejor funcionamiento; así como elementos añadidos establecidos según las necesidades surgidas en el proceso de desarrollo.

- El primer cambio que se realizó en el sistema, fue la incorporación del componente encargado de la comunicación entre el procesador Arduino y el PC (FTID). Este no se contemplaba en un inicio, no fue hasta que se comprobó que el procesador Arduino no disponía de conexión directa entre él y un PC cuando se empezaron a estudiar las diferentes opciones.

  Se valoraron varias posibilidades de conexión, al final se decidió adquirir este tipo de componente por su facilidad de conexión y su seguridad ante posibles fallos. Así se consiguió la alimentación del sistema y su transferencia de datos con el PC. Una vez terminado el sistema, este módulo resulta innecesario ya que el sistema puede ser alimentado utilizando una batería.

- La primera idea para simplificar el circuito fue soldar los componentes a una placa vacía, trazando caminos de soldadura entre los diferentes pines logrando conectarlo entre sí. Esta idea fue sustituida por la realización de una placa mediante la aplicación Kicad, con esto se consigue diseñar un sistema más firme y seguro evitando el uso de cables para las conexiones y facilitando el encapsulado.

- Durante el diseño de la placa en Kicad se estudiaron distintas formas de conectar los componentes entre ellos, debido a la cantidad de conexiones resultaba difícil encajar todo de forma que todas se encontraran en la misma cara sin que hubiera cortes entre ellas. La opción escogida para solucionar estos problemas fue conectar todas las conexiones a tierra a un plano situado en la cara superior de la placa y rodea el resto del circuito, es lo que permitió conectar el resto de conectores en la misma cara sin cortes entre ellos.

- En un principio, estaba pensado que los componentes serían soldados directamente a la placa. Esto se modificó más tarde cuando se decidió que se soldarían a la placa pines encargados de conectar los componentes a la placa, permitiendo la conexión y desconexión de los mismos, además de añadir la posibilidad de realizar cambios si hubiese algún fallo o sustituirlos por unos nuevos.

- Introducir el sistema en un encapsulado. Este añadido al sistema sirve para aislarlo completamente de los impactos externos, tanto de las temperaturas como los efectos provocados por los diferentes tipos de luz cuando se realicen adquisiciones. También otorga la ventaja de hacer el sistema más transportable sin poner en peligro los componentes en el trayecto.

- Desde el comienzo del proyecto la forma de alimentar el circuito ha sido a través de un PC mediante un cable USB. Esto resulta útil, pero, si se pretende utilizar el sistema de forma portátil, es necesario poder alimentarlo sin tener un PC cerca. Se estudiaron varias formas de alimentar el sistema Arduino, la opción escogida fue una pila de 9V conectada a los pines Vcc y GND del Arduino. Se eligió esta opción por ser la más sencilla y fácil de implementar en el sistema ya diseñado.

Todas estas consideraciones han concluido en un mejor funcionamiento y una simplificación del sistema, mejorando su diseño.

3.2. Software

En este apartado se redactará el proceso de desarrollo del software del sistema utilizando el mismo procedimiento llevado a cabo anteriormente con el hardware. El software diseñado para el proyecto se compone de varias partes y resulta una parte compleja y grande del proyecto en general.

Cabe destacar que al comienzo del proyecto los conocimientos acerca de los lenguajes de programación pertenecientes a Arduino [26] y Android [27] resultaban nulos. Por ello se dedicó tiempo al estudio y preparación de estos lenguajes para poder realizar el proyecto lo mejor posible utilizando los manuales encontrados que hacían referencia a los conocimientos que se pretendían aplicar.

Antes de comenzar a explicar el proceso por el cual se ha realizado el software del sistema, es necesario destacar que esto se desarrollará dividiendo el apartado en dos partes

importantes. La primera dedicada a la explicación del software realizado para el sistema Arduino, encargado de la adquisición de datos su tratamiento y envío. La segunda parte desarrollará la explicación del software realizado para el sistema Android, donde se recibirán los datos y estos serían tratados o almacenados.

3.2.1. Software Arduino

El software [28] diseñado para la parte del sistema controlada por el procesador Arduino fue la primera en crearse, ya que es la encargada de tomar las adquisiciones en un inicio.

Se basa en permitir la adquisición del color mediante el detector conectado al procesador Arduino, procesar estos datos y enviarlos a través de bluetooth hasta el dispositivo móvil. A continuación, se presenta una explicación detallada de estos pasos y los cambios que se fueron realizando durante su desarrollo.

Para la realización de este apartado se utilizó la aplicación ”Arduino Studio”, que se emplea para programar el procesador Arduino y transmitir estos programas a su memoria interna, lo que permite ejecutarlos sin necesidad de estar conectado a un PC.

Tras estudiar la hoja de características propia del detector, se realizó su conexión al sistema Arduino. Para la comunicación, se utilizó una biblioteca ya establecida para procurar el buen funcionamiento del detector. Esta se encargaba de enviar las señales necesarias al detector con el tiempo necesario para obtener a la salida los valores del color detectado, figura 3.9. En un comienzo fue elaborada una biblioteca personal encargada de acceder al detector y hacerlo funcionar correctamente, pero esta biblioteca fue descartada tras un tiempo debido a su gran similitud con la biblioteca ya elaborada. Ambas estaban basadas en las mismas clases y métodos, diferenciándose en algunas ocasiones exclusivamente por el nombre dado.

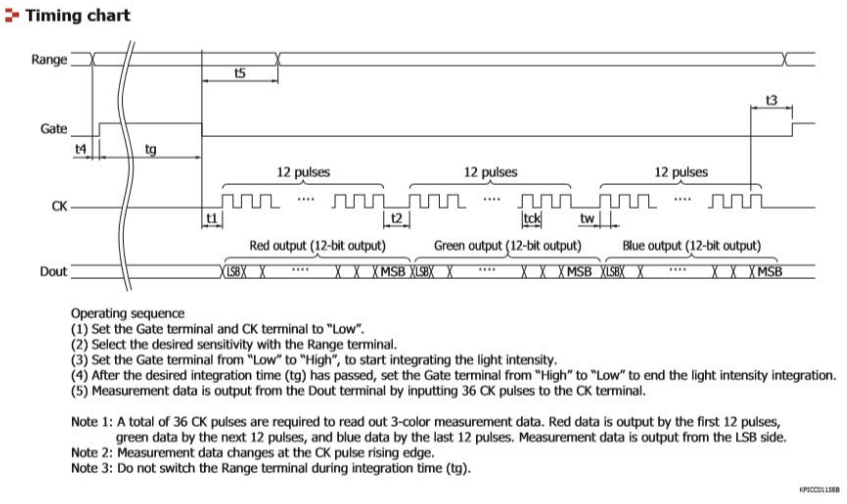


Fig. 3.9. Información de los pasos a seguir para obtener valores de la medida [13]

A la biblioteca utilizada se le realizaron varios cambios que permiten modificar partes importantes del código desde el código base sin tener que acceder a esta. Un ejemplo de esto es el poder modificar el número de píxeles del detector que se utilizan en cada medida. Por defecto en el proyecto se utiliza siempre el máximo de píxeles, pero se generó la opción de modificarlo rápidamente si se quisiera. Esto se hizo extrayendo de la biblioteca el proceso que obtenía el valor del número de píxeles a utilizar y permitiendo introducir por parámetros este valor.

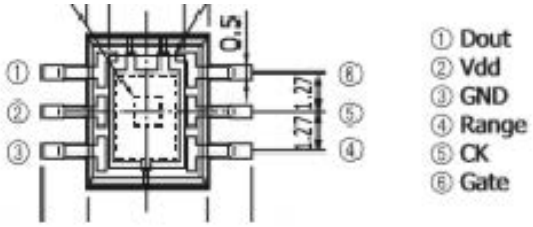


Fig. 3.10. Pines de conexión del detector [13].

Una vez establecida la conexión con el detector, los datos eran recibidos y guardados en variables establecidas. Estos datos debían ser tratados, ya que se obtenía la medida de los tres colores en una cadena de 36 dígitos binaria. Dividido en tres partes, cada 12 dígitos equivale al valor medido de un color primario (Rojo, Verde y Azul). Separando el valor recibido y convirtiéndolo a un valor numérico con una operación sencilla se obtiene el valor final medido de cada uno de los colores.

El siguiente paso era establecer la conexión bluetooth para poder enviar estos valores. Para ello se realizaron varias pruebas y se establecieron dos pines del procesador Arduino como TX y RX para permitir su comunicación con la antena bluetooth. Se utilizó una aplicación ya establecida para conexiones bluetooth con Arduino, se realizaron las pruebas del establecimiento de conexión y envío de datos. Finalmente se consiguió recibir los valores de cada medida en el teléfono móvil, que disponía de esta aplicación.

Aprovechando estos logros, se modificó el código para establecer un indicador de inicio de medida, una señal encargada de iniciar la medida con el detector. Se alcanzó el envío de datos en la dirección inversa, desde la aplicación al Arduino. De esta forma la adquisición se iniciaba cuando un botón específico es pulsado en la aplicación.

Con esto conseguido, se procedió a pulir el código de la aplicación y mejorar los resultados. Finalmente, el valor obtenido tras cada adquisición viene representado de la siguiente forma ”RXXXGXXXBXXX” Donde las ”X” representan los valores numéricos pertenecientes a cada color. Este valor es enviado por la antena bluetooth al dispositivo que esté conectado a ella. Tras cada medida un aviso es enviado al teléfono indicando que puede realizarse una medida nueva pulsando nuevamente el botón.

```
int red = colorSensor.getRed();
int green = colorSensor.getGreen();
int blue = colorSensor.getBlue();

 BTSerial.print("R");
 BTSerial.print(red);

 BTSerial.print("V");
 BTSerial.print(green);

 BTSerial.print("A");
 BTSerial.print(blue);
 BTSerial.print("#");
 //BTSerial.println("");
```

Fig. 3.11. Código que envía los valores de la medida por bluetooth.

El siguiente paso fue diseñar la aplicación Android encargada de recibir estos valores y tratarlos, llamada ”Detector Color”.

3.2.2. Software Android

Como el sistema diseñado para la aplicación ”Detector Color” resulta bastante complejo de explicar, se realizará de la siguiente manera: primero se realizará una explicación exhaustiva de los procesos que realiza la aplicación y el funcionamiento de su interfaz, así como un ”tutorial” de uso para comprender las posibilidades que ofrece. Una vez se haya comprendido su funcionamiento, se procederá a explicar en más detalle los procesos de programación [29] necesarios, haciendo hincapié en los más complejos o que llevarán mayor complicación.

Interfaz de la aplicación ”Detector Color”

Para la aplicación desarrollada fueron diseñadas cuatro pantallas o actividades (nombre dado por el software de programación), con las cuales se realizarán los procedimientos necesarios con los datos de los colores.

La primera pantalla se encuentra nada más encender la aplicación, encargándose primeramente de comprobar si está conectado el bluetooth en el teléfono. En caso de no estar conectado, saltará un mensaje preguntando si se quiere conectar. En caso de responder "sí", la conexión bluetooth se establecerá y aparecerá en pantalla la lista de dispositivos vinculados al teléfono, por lo que es necesario tener vinculada la antena bluetooth del Arduino previo a su uso. Después, se seleccionará el sistema y, una vez establecida la conexión, pasará a la siguiente pantalla.

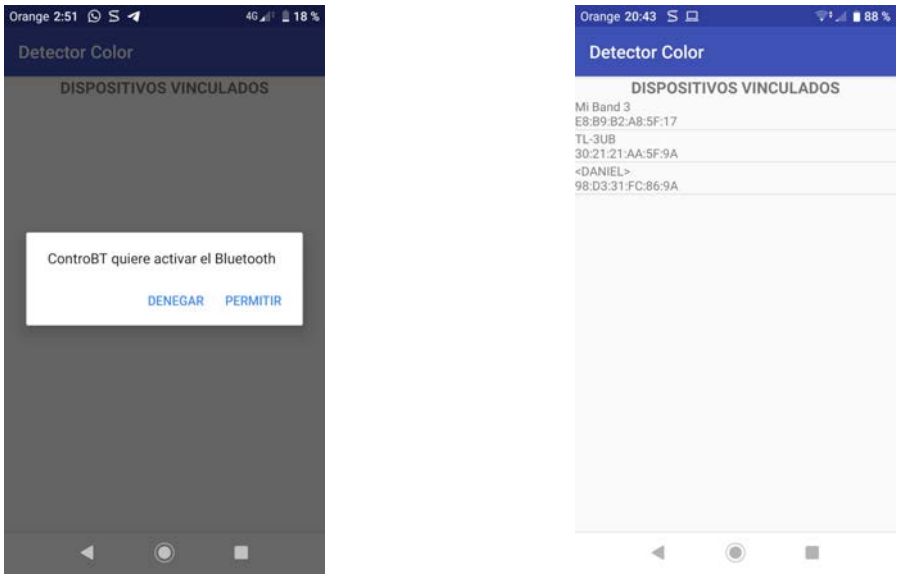
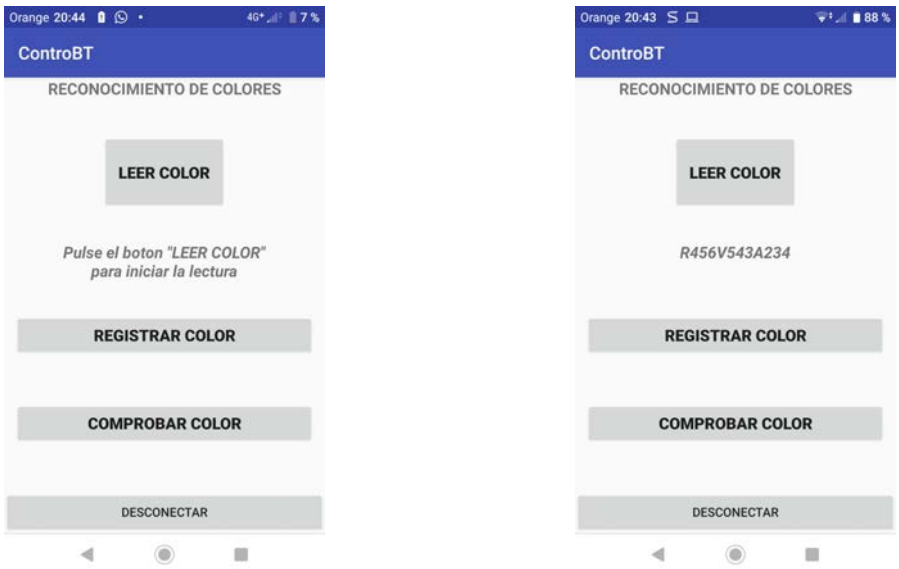


Fig. 3.12. Primera pantalla de la aplicación, pueden verse los dos pasos a seguir.

Esta segunda pantalla será la principal del sistema donde, aparecen tres opciones: Leer color, Registrar Color y Comprobar Color.

El botón que será pulsado en primer lugar será ”Leer Color”, tras lo que este enviará una señal al sistema Arduino, que iniciará la medida del color. Una vez finalizada, se recibirán los datos del color medido en forma de cadena de caracteres, como, por ejemplo: R245V234A12, equivaliendo a los siguientes valores: Rojo: 245, Verde: 234, Azul: 12. Este valor aparecerá en la pantalla del teléfono, pudiendo visualizar inicialmente los datos obtenidos esto permitirá tomar una nueva medida antes de realizar ninguna otra tarea con los datos obtenidos si el valor hace pensar que algún error ha ocurrido y la medida ha sido errónea.



(a) Sin medida.

(b) Con medida.

Fig. 3.13. Pantalla principal

Una vez realizada la medida, existen dos opciones, registrar color o comprobar color. Cualquiera de estas opciones abrirá una nueva ventana en la aplicación.

Pulsando la opción ”Registrar Color” pasará a la pantalla, encargada de registrar el color en la base de datos diseñada. Lo primero que puede verse es un desglose de los datos obtenido en la medida, divididos ya en los tres colores medidos: rojo, verde y azul.

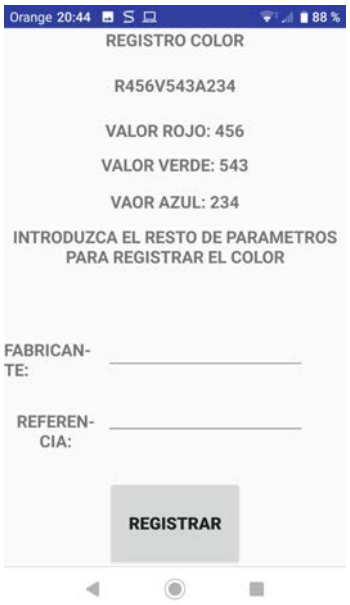


Fig. 3.14. Pantalla encargada del registro de los colores.

En esta pantalla se pide que el usuario que introduzca los valores ”Fabricante” y ”Número de registro” del color medido. Una vez introducidos, al pulsar la opción ”Registrar color”, el color queda guardado en la base de datos, apareciendo un mensaje que muestra el número de colores registrados previamente. Tras realizar esta acción, pulsando el botón ”atrás” del teléfono móvil, se vuelve a la pantalla principal para poder realizar una nueva medida si fuera necesario.

Una vez realizada la medida, en caso pulsar el botón ”Comprobar Color” la aplicación cambiará a una nueva pantalla. Al iniciarse realizará el mismo proceso explicado anteriormente, se mostrarán por pantalla los valores individuales de cada color medidos por el detector. Además, aparecerá un desplegable o ”Spinner” donde se puede escoger el nombre del fabricante con quien se quiere comparar la medida hecha. Al realizar la comparación, la aplicación mostrará por pantalla el número de referencia y nombre del fabricante del color más semejante al medido.

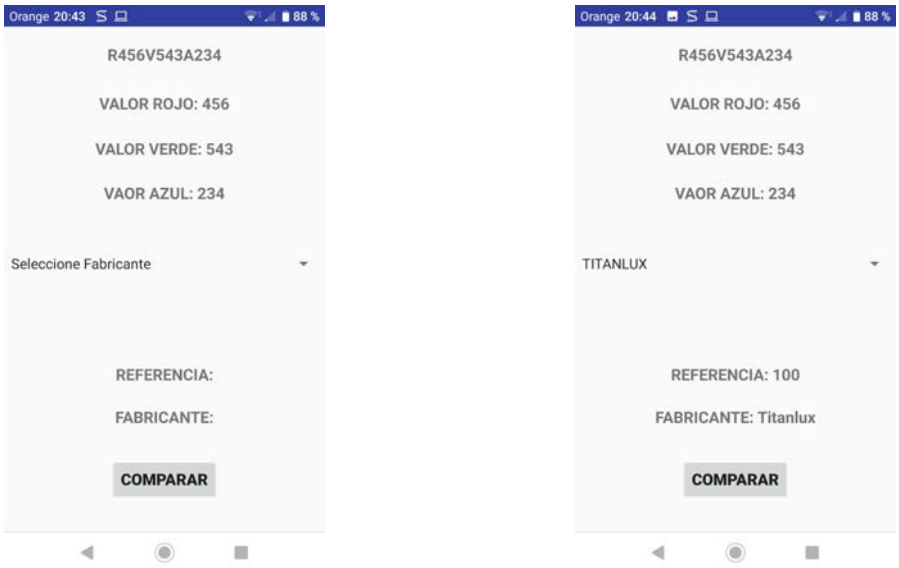


Fig. 3.15. Pantalla encargada de la comparación de colores. Se muestra la pantalla antes y después de realizar una comparación.

Con esto queda explicado el uso e interfaz de la aplicación. A continuación, se explicarán más detalladamente los procesos de programación llevados a cabo para obtener estos resultados.

Programación del software

Para la programación de la aplicación ”Detector Color” se utilizó la aplicación propia de Android, ”Android Studio”, diseñada por Google y encargada de programar su sistema operativo.

A continuación, se explican los procesos de programación llevados a cabo para conseguir los resultados esperados en cada una de las pantallas establecidas.

- La programación realizada en la primera pantalla que se encuentra en la aplicación, 3.12, se basa en, habilitar la conexión con bluetooth del teléfono y acceder a los datos de dispositivos vinculados para generar la lista de dispositivos. Android ofrece estas posibilidades utilizando una serie de comandos propios del sistema operativo. La parte más compleja fue encontrar los comandos necesarios que realizaban las tareas requeridas: tanto establecer las conexiones como los permisos necesarios.

```
private void VerificarEstadoBT() {
    // Comprueba que el dispositivo tiene Bluetooth y que está encendido.
    mBtAdapter= BluetoothAdapter.getDefaultAdapter();
    if(mBtAdapter==null) {
        Toast.makeText(getBaseContext(), text: "El dispositivo no soporta Bluetooth", Toast.LENGTH_SHORT).show();
    } else {
        if (mBtAdapter.isEnabled()) {
            Log.d(TAG, msg: "...Bluetooth Activado...");
        } else {
            //Solicita al usuario que active Bluetooth
            Intent enableBtIntent = new Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
            startActivityForResult(enableBtIntent, requestCode: 1);

        }
    }
}
```

Fig. 3.16. Fragmento de código que activa la conexión bluetooth del teléfono

La lista se completa con los datos recogidos de los dispositivos vinculados. Se estableció una función que se inicia cuando un dispositivo de la lista es seleccionado. Tras escoger el dispositivo a conectar, pasa a la siguiente pantalla enviando la dirección MAC del dispositivo escogido.

```
@Override
public void onResume()
{
    super.onResume();
    //----------------------------------
    VerificarEstadoBT();

    // Inicializa la array que contendra la lista de los dispositivos bluetooth vinculados
    mPairedDevicesArrayAdapter = new ArrayAdapter( context: this, R.layout.nombre_dispositivos);
    // Presenta los disposisitivos vinculados en el ListView
    IdLista = (ListView) findViewById(R.id.IdLista);
    IdLista.setAdapter(mPairedDevicesArrayAdapter);
    IdLista.setOnItemClickListener(mDeviceClickListener);
    // Obtiene el adaptador local Bluetooth adapter
    mBtAdapter = BluetoothAdapter.getDefaultAdapter();
    // Obtiene un conjunto de dispositivos actualmente emparejados y agregua a 'pairedDevices'
    Set <BluetoothDevice> pairedDevices = mBtAdapter.getBondedDevices();
    // Adiciona un dispositivos previo emparejado al array
    if (pairedDevices.size() > 0)
    {
        for (BluetoothDevice device : pairedDevices) {
            mPairedDevicesArrayAdapter.add(device.getName() + "\n" + device.getAddress());
        }
    }
}
```

Fig. 3.17. Fragmento de código que genera la lista de dispositivos vinculados al teléfono.

- La primera acción del sistema en la pantalla encargada de la lectura del color (figura 3.13) es establecer la conexión con el socket bluetooth del sistema escogido anteriormente, sirviéndose de su dirección MAC y permitiendo el envío de datos inalámbricamente. Se utiliza el servicio SPP UUID para establecer una salida de datos segura a través de bluetooth.

  A modo de seguridad, se programó una función encargada de no dejar abierto el socket cuando la aplicación salga de esta pantalla.

  El botón encargado de iniciar la medida del color se programó de tal forma que, al pulsar se envía un valor lógico por bluetooth, iniciando así la medida en el sistema

Arduino. A su vez, otra función se encargará de revisar constantemente si algún valor es recibido a través de bluetooth desde el Arduino (figura 3.18), siendo guardado en la variable que se muestra en la pantalla de la aplicación.

```
bluetoothIn = new Handler() {
    public void handleMessage(android.os.Message msg) {
        if (msg.what == handlerState) {
            String readMessage = (String) msg.obj;
            DataStringIN.append(readMessage);

            int endOfLineIndex = DataStringIN.indexOf("#");

            if (endOfLineIndex > 0) {
                String dataInPrint = DataStringIN.substring(0, endOfLineIndex);
                IdBufferIn.setText(dataInPrint);
                DataStringIN.delete(0, DataStringIN.length());
            }
        }
    }
};
```

Fig. 3.18. Imagen código correspondiente a la lectura de los datos recibidos.

Al pulsar cualquiera de los otros dos botones, ”Registrar Color” y ”Comprobar Color”, se inicia la función encargada de cambiar la pantalla de la aplicación, enviando el valor de la medida, ya que será utilizado en la siguiente pantalla.

- En la pantalla ”Registrar Color” (figura 3.14), la primera acción que realiza el software es procesar el valor de la medida recibido. Este valor se recibe como una cadena de datos, está formada por los valores de los tres colores medidos por el detector, encontrando al inicio de cada valor la letra mayúscula del color al que pertenece. La cadena es separada mediante un bucle encargado de evaluar uno a uno los valores que componen la cadena y dividirlos en tres variables diferentes.

```
String dato_color = extras.getString( key: "COLOR");
IdDato.setText(dato_color);
for (int i = 0; i < dato_color.length(); i++){
   char x=dato_color.charAt(i);
    if(x == 'V') { g = i; }
    if(x == 'A') { b = i; }
}
String rojo = dato_color.substring(1,g);
String verde = dato_color.substring(g+1,b);
String azul = dato_color.substring(b+1,dato_color.length());

IdR.setText("VALOR ROJO: " + rojo);
IdA.setText("VAOR AZUL: " + azul);
IdV.setText("VALOR VERDE: " + verde);

Drojo = Integer.parseInt(rojo);
Dverde = Integer.parseInt(verde);
Dazul = Integer.parseInt(azul);
```

Fig. 3.19. Imagen código correspondiente a la división del valor medido en los diferentes colores.

Las variables que contienen los valores individuales de los colores son las mostradas por pantalla. Al pulsar el botón ”Registrar Color” el software captura en otras variables los datos que el usuario ha introducido por pantalla y, junto con los valores de la medida, son almacenados en la base de, este proceso se explica más adelante en el apartado correspondiente a la base de datos.

```
public void onClick(View view){
    registrarColores();
}

private void registrarColores() {
    ConexionSQLiteHelper conn=new ConexionSQLiteHelper( context: this, name: "bd_colores", factory: null, version: 1);

    SQLiteDatabase db=conn.getWritableDatabase();
    ContentValues values = new ContentValues();
    values.put(Utilidades.CAMPO_ROJO,Drojo);
    values.put(Utilidades.CAMPO_VERDE,Dverde);
    values.put(Utilidades.CAMPO_AZUL,Dazul);
    values.put(Utilidades.CAMPO_REFERENCIA,IdReferencia.getText().toString());
    values.put(Utilidades.CAMPO_FABRICANTE,IdFabricante.getText().toString());

    Long idResultante=db.insert(Utilidades.TABLA_COLORES,Utilidades.CAMPO_REFERENCIA,values);

    Toast.makeText(getApplicationContext(), text: "Id Registro: " + idResultante, Toast.LENGTH_SHORT).show();

    db.close();

}
```

Fig. 3.20. Fragmento de código encargado de registrar un color en la base de datos.

- El software, al iniciarse la pantalla ”Comparar Color”, (figura 3.15) realiza el mismo procedimiento que la pantalla anterior, dividiendo los valores de la cadena de la medida en diferentes variables. Una vez hecho esto, el software está programado para acceder a la base de datos generada, utilizando los comandos que proporciona Android. Con el acceso establecido a la base de datos, una función bucle se encarga de repasar uno a uno los datos de las medidas guardadas anteriormente. Con cada medida encontrada en la base de datos, se crea un nuevo objeto de tipo ”Color”, clase creada especialmente para este proceso, cuyos atributos se componen de los datos guardados en cada medida. A su vez, se genera un vector que almacena, en orden, los objetos color creados, de esta manera resulta sencillo y rápido acceder a los colores registrados y a sus características.

```
private void consultarListaColores() {

    SQLiteDatabase db=conn.getReadableDatabase();
    Color color = null;
    coloresList =new ArrayList<Color>();

    Cursor cu = db.rawQuery( sql: "SELECT * FROM " +Utilidades.TABLA_COLORES , selectionArgs: null);

    while(cu.moveToNext()){
        color = new Color();
        color.setRojo(cu.getInt( i: 0));
        color.setVerde(cu.getInt( i: 1));
        color.setAzul(cu.getInt( i: 2));
        color.setReferencia(cu.getString( i: 3));
       color.setFabricante(cu.getString( i: 4));

        coloresList.add(color);
   }
    obtenerFabricantes(); //nuevo
}
```

Fig. 3.21. Imagen código encargado de almacenar los colores de la base de datos en un vector.

Una vez generado el vector de colores el siguiente paso del software es establecer la lista de fabricantes con los que se puede comparar la medida. Se ha programado un desplegable seleccionable que contiene los nombres de todos los fabricantes guardados en la base de datos. Este desplegable o ”Spinner”, como es llamado en el lenguaje de programación de Android, se genera automáticamente cada vez que la aplicación entra en esta pantalla. De esta manera, si han sido registrados colores de un nuevo fabricante, este aparecerá en el desplegable cuando se quiera realizar una comparación.

Este proceso de actualización automático se realiza mediante una función bucle que recorre todo el vector de colores (figura 3.22), en cada iteración se evalúa el atributo ”Fabricante” de cada color. La primera operación que realiza el bucle es, establecer las letras del nombre del fabricante en minúscula, de esta forma si un usuario ha escrito el nombre de un fabricante de diferente forma, el software lo tomará como el mismo fabricante y no se incluirá dos veces en la lista.

Una vez realizada esta operación se genera otro bucle encargado de comparar el nombre del fabricante seleccionado, y en minúscula, con los nombres ya guardados en la lista. Si durante esta comparación el programa encuentra un fabricante idéntico ya escrito en el desplegable, el fabricante comparado es descartado y se pasa a la siguiente iteración. En el caso contrario, donde no se encuentra ninguna similitud con los nombres del desplegable, el programa añade el nombre del fabricante a la lista.

```
private void obtenerFabricantes() {
    Lista_Fabricantes= new ArrayList<String>();
    Lista_Fabricantes.add("Seleccione Fabricante");
    Lista_Fabricantes.add("Todos");
    String fabri;
    String fab;
    for(int q=0;q<coloresList.size();q++){
        fabri = coloresList.get(q).getFabricante();
        String uperfabri = fabri.toUpperCase();
        for(int w=0;w<Lista_Fabricantes.size();w++ )
        {

            fab = Lista_Fabricantes.get(w);
            String uperfab = fab.toUpperCase();
            if(uperfabri.equals(uperfab))
            {

                s++;
            }

        }

        if(s == 0) {
            Lista_Fabricantes.add(coloresList.get(q).getFabricante().toString().toUpperCase());

        }
        s = 0;
    }


}
```

Fig. 3.22. Código que genera el listado de fabricantes.

Una vez se ha generado el desplegable, el usuario al escoger uno de ellos e iniciar la comparación, el software realiza otro bucle, en el cual, cada iteración compara los valores del color medido con los valores de cada color del vector correspondientes al fabricante seleccionado. Para realizar esto, se evalúan los nombres de los fabricantes de cada color, asegurando la comparación entre colores pertenecientes al mismo fabricante escogido, utilizando el mismo procedimiento que se utiliza para generar el desplegable. La comparación entre colores se realiza utilizando la fórmula 3.1

$$Comparacion = \sqrt{(R_{ad} - R_{ref})^2 + (V_{ad} - V_{ref})^2 + (A_{ad} - A_{ref})^2} \qquad (3.1)$$

Tras las operaciones, el color con el que se obtenga el valor más pequeño de la variable ”Comparación” es el color que más se aproxima al medido, y, una vez finalizada la comparación, queda registrada la posición del vector en la que se encuentra el color más parecido. Lo que permite mostrar por pantalla los atributos ”Nombre de Fabricante” y ”Número de Referencia” del mismo.

```
IdComparar.setOnClickListener(new View.OnClickListener(){
    @Override
    public void onClick(View v)
    {


        compararvalores();


       IdReferencia.setText("REFERENCIA: " + coloresList.get(posicionColor).getReferencia());
       IdFabricante.setText("FABRICANTE: " + coloresList.get(posicionColor).getFabricante());

       x = 10000;

    }
});
```

Fig. 3.23. Fragmento de código que muestra los datos del Fabricante y la Referencia del color más parecido tras la comparación.

3.2.3. Requisitos

Uno de los requisitos del sistema a tener en cuenta por el usuario antes de utilizar la aplicación, es haber vinculado al teléfono móvil la señal bluetooth del sistema Arduino. Sin vincularlo previamente, es imposible realizar la conexión entre la aplicación y el sistema.

Otro requisito indispensable pertenece al apartado de comparación de color, donde es necesario seleccionar una de las opciones establecida para la comparación. Si ninguna opción fuera seleccionada, la comparación no se realizaría correctamente.

Finalmente, a la hora de realizar varias adquisiciones seguidas, es necesario esperar un tiempo prudente para realizar otra adquisición ya que el programa espera unos segundos antes de iniciar el proceso de adquisición otra vez.

3.2.4. Código del sistema (diagrama de estados)

Se presenta un diagrama UML del funcionamiento del sistema. Este diagrama expresa el funcionamiento del sistema paso a paso incluyendo las opciones del usuario y los procesos que ejecuta el sistema.

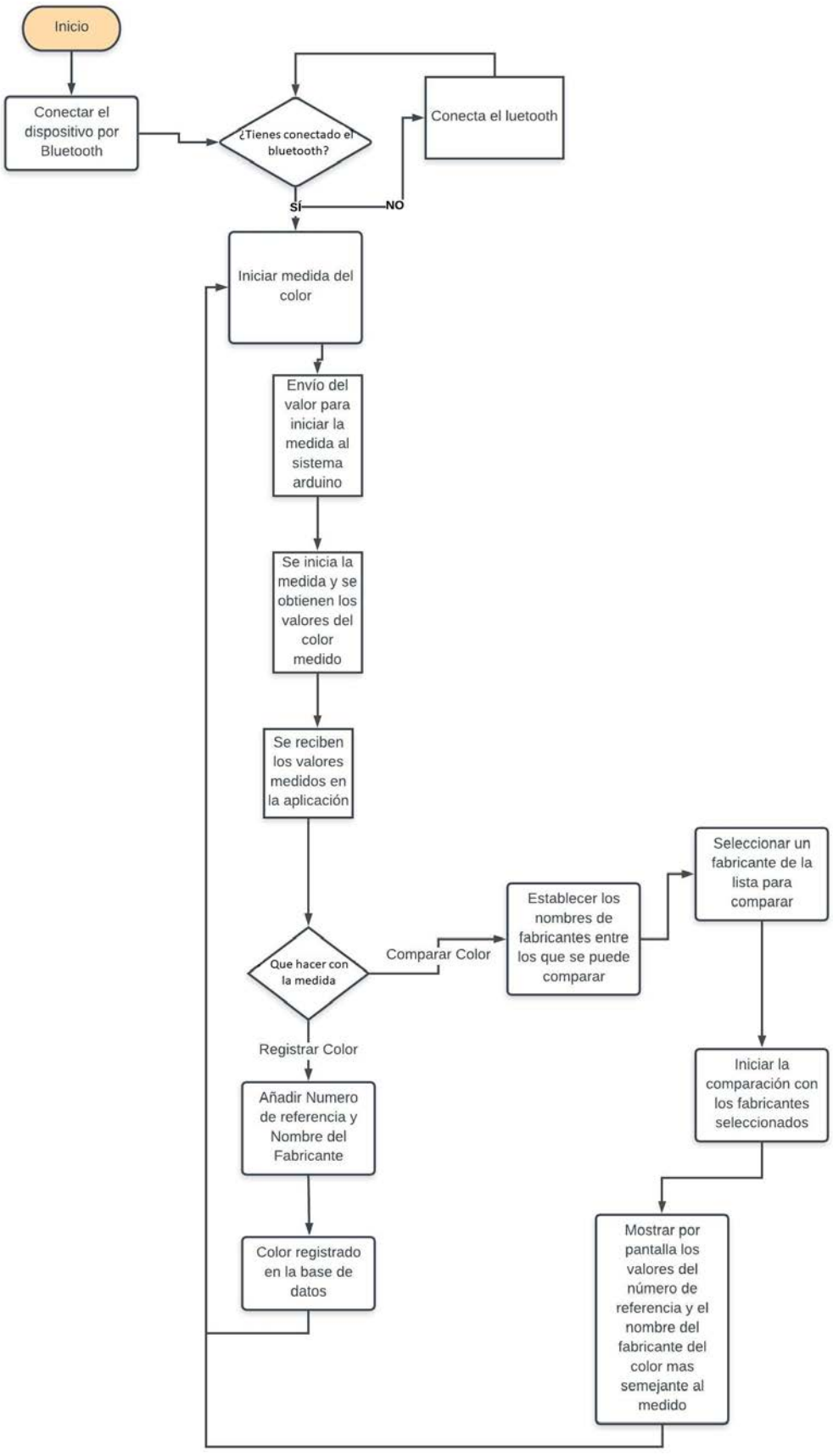


Fig. 3.24. Diagrama de Estados.

3.2.5. Base de datos

En este apartado se procede a explicar, de la manera más detallada posible, el desarrollo de la base de datos utilizada para registrar los diferentes colores medidos en la aplicación. Para la realización de esta base de datos se utiliza SQLite ya que este motor se encuentra incluido en Android y posee todas las herramientas necesarias para poder trabajar con ella.

Lo primero que se realiza es la estructura que tendrá la base de datos, estará definida por las características registradas de cada color: Nº de Referencia, Fabricante, Valor Rojo, Valor Verde y Valor Azul. El siguiente paso es generar una nueva clase que servirá para conectar la aplicación con la base de datos. En esta clase se añade el método constructor que genera la base de datos SQLite y el método encargado de actualizar la base de datos. Este, comprueba cada vez que se crea de nuevo la base de datos si ya existe, y en caso afirmativo, esta se actualiza con los nuevos datos introducidos, esto es útil para prevenir que, cada vez que se actualice la aplicación en el dispositivo móvil, no se pierdan los datos registrados.

```
import android.content.Context;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;

import com.estudiantes.controbt.utilidades.Utilidades;

public class ConexionSQLiteHelper extends SQLiteOpenHelper {


    public ConexionSQLiteHelper(Context context, String name, SQLiteDatabase.CursorFactory factory, int version) {
        super(context, name, factory, version);
    }

    @Override
    public void onCreate(SQLiteDatabase db) {
        db.execSQL(Utilidades.CREAR_TABBLA_COLOR);

    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int version_antigua, int version_nueva) {
        db.execSQL("DROP TABLE IF EXISTS colores");
        onCreate(db);
    }
}
```

Fig. 3.25. Imagen código conexión base de datos SQL.

Hasta aquí se ha conseguido generar la base de datos en el sistema, el siguiente paso es conseguir registrar en esa base de datos, los valores de cada color medido. Para ello y para facilitar la programación, se genera una clase encargada de almacenar valores constantes, estas constantes representan los campos que posee la base de datos.

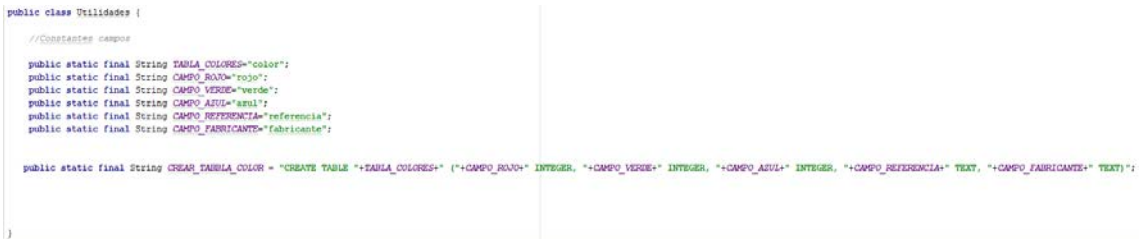


Fig. 3.26. Fragmento de código donde se muestran las constantes utilizadas en la base de datos.

En la pantalla encargada del registro de colores, se capturan en variables los valores del color medido que se añadirán a la base de datos. Tras esto, al pulsar el botón ”Registrar Color” se llamará a un método que inicialmente abra la conexión a la base de datos para poder guardar la estructura en ella, con una serie de comandos propios del sistema, y utilizando las constantes establecidas en la clase creada, se añaden a la base de datos los valores del color medido.

```
private void registrarColores() {
    ConexionSQLiteHelper conn=new ConexionSQLiteHelper( context: this, name: "bd_colores", factory: null, version: 1);

    SQLiteDatabase db=conn.getWritableDatabase();
    ContentValues values = new ContentValues();
    values.put(Utilidades.CAMPO_ROJO,Drojo);
    values.put(Utilidades.CAMPO_VERDE,Dverde);
    values.put(Utilidades.CAMPO_AZUL,Dazul);
    values.put(Utilidades.CAMPO_REFERENCIA,IdReferencia.getText().toString());
    values.put(Utilidades.CAMPO_FABRICANTE,IdFabricante.getText().toString());

    Long idResultante=db.insert(Utilidades.TABLA_COLORES,Utilidades.CAMPO_REFERENCIA,values);

    Toast.makeText(getApplicationContext(), text: "Id Registro: " + idResultante, Toast.LENGTH_SHORT).show();

    db.close();

}
```

Fig. 3.27. Fragmento del código encargado de registrar un color.

Con esto se genera la tabla almacenada en la base de datos y se registran los colores en ella. Para acceder a los colores se sigue la misma estructura, primero hay que habilitar la conexión con la base de datos, después, utilizando el comando adecuado, se accede a sus datos guardando en un vector cada objeto, de tipo color, almacenado en ella.

3.3. Escoger el tiempo de integración (IT)

Una vez se ha finalizado el diseño del sistema, es necesario establecer un tiempo de integración para el detector. El tiempo de integración escogido debe permitir los siguientes requisitos: El sistema debe tener una buena sensibilidad, el sistema no debe saturar en ninguna medida y el sistema tiene que ser capaz de medir, aunque el color sea muy oscuro.

Esto se consigue realizando medidas a la par que se va cambiando el tiempo de integración. En la práctica, se ha buscado un tiempo que permita medir un objeto totalmente blanco y reflectante sin saturar y, a la vez medir un objeto negro y seguir teniendo una me-

dida válida. Tras muchas pruebas se ha concluido que el IT más acorde a las necesidades del sistema es 900ms.

4. RESULTADOS

Tras completar el sistema, el siguiente paso es comenzar a realizar las primeras adquisiciones que serán tomadas como referencia a la hora de comparar los diferentes colores.

4.1. Prueba 1 - Escoger fondo.

Para las primeras pruebas se han utilizado paletas de colores de los fabricantes ”Titanlux” y ”Bruguer”. Estas paletas se han obtenido a través de revistas promocionales o internet, imprimiendo paletas completas de colores. Esto supone un problema para la adquisición, ya que el papel resulta muy fino y provoca una transmisión de intensidad a través de él, provocada por la luz ambiente. Una posible solución al problema es colocar, detrás del papel, un objeto opaco impidiendo la transmisión. Se dudó entre colocar un objeto totalmente blanco, por ejemplo, un papel doblado varias veces, y un objeto que fuera negro. Se realizaron pruebas con ambos objetos de fondo, los resultados de estas pruebas están expuestos en la siguiente tabla.

Fondo Negro		
R	**G**	**B**
539	431	364
1582	2648	2345
1056	1722	1664
779	1257	1321
699	1034	1040
1459	2262	1936

Fondo Blanco		
R	**G**	**B**
561	436	369
1715	2823	2541
1140	1829	1797
820	1308	1401
737	1077	1099
1586	2413	2091

TABLA 4.1. TABLA COMPARACIÓN ENTRE FONDOS

Como se puede observar, las medidas tomadas con fondo blanco otorgan un nivel digital ligeramente aumentado frente a las medidas tomadas con fondo negro. Esto se debe a la transmisión de la luz a través del papel de medida. Al colocar un fondo detrás del papel se ha evitado la transmisión de la luz ambiente hacia el detector, pero todavía existe la transmisión de la luz provocada por el led situado dentro del sistema. Esta luz se transmite a través del papel y termina en el fondo colocado. En el caso del fondo negro toda la luz es absorbida por este, y en el caso del fondo blanco, la luz se refleja, siendo transmitida nuevamente hacia el interior del sistema, generando ese aumento de intensidad que puede observarse en las medidas. Es un aumento leve, ya que en todo momento los colores siguen guardando sus características propias, y a la hora de la comparación no supondría un problema. A pesar de ello, a partir de este punto, se decidió tomar el resto de medidas utilizando el fondo negro.

4.2. Prueba 2 - Comprobar sensibilidad del sistema.

Estas pruebas están enfocadas en conocer la sensibilidad y fiabilidad del sistema. Se realizan varias repeticiones de cada medida para comprobar la desviación de los datos en cada una de ellas. A continuación, se expone una tabla con los datos de estas medidas recogidas.

COLOR	R	G	B
	539	431	364
	539	432	364
	540	433	365
	540	433	366
	539	432	365
	668	406	391
	669	407	391
	669	408	392
	669	408	392
	669	408	392
	1588	690	473
	1591	692	475
	1591	693	475
	1591	693	475
	1592	693	476
	1368	2655	763
	1369	2656	763
	1370	2658	764
	1370	2657	764
	1370	2659	764
	1044	2350	2607
	1045	2352	2609
	1045	2353	2611
	1045	2353	2611
	1046	2354	2612

TABLA 4.2. TABLA COMPARACIÓN ENTRE MEDIDAS

Como se observa en la tabla, las medidas sufren una variación mínima entre ellas. Con estos datos se estima que el sistema funciona correctamente y se da paso al registro de los diferentes colores obtenidos de las marcas ”Titanlux” y ”Bruguer”. A continuación, se expone en una tabla los colores registrados en la aplicación junto con las referencias dadas por el fabricante.

TITANLUX			
NÚMERO REF	**R**	**G**	**B**
504	1578	240	2339
509	1058	1727	1670
503	779	1257	1321
510	699	1034	1041
508	1459	2262	1938
555	539	531	365
523	669	408	392
563	848	446	457
564	1013	473	442
554	1588	690	473
514	281	772	543
522	1368	2655	763
516	343	1166	695
512	688	1822	1358
538	1266	2632	1759
542	287	520	740
539	258	869	1973
536	234	1053	2342
535	691	1806	2420
540	1046	2354	2612
586	1859	2875	1647
527	1843	2729	1412
532	1826	2609	730
529	1860	2177	719
568	1520	1653	589

BRUGUER			
NUMERO REF.	**R**	**G**	**B**
01080	1928	3055	2078
61575	1940	2868	1841
22565	1704	2289	1271
22555	1413	1893	1113
60580	1989	3192	2418
80570	1648	2517	1993
40565	1499	2134	1674
41040	811	1220	965
40520	563	758	642
00050	1023	1552	1314
00069	1537	2477	2136
00081	1888	3149	2611
00585	1835	3361	3014
01070	1386	2593	2522
03030	546	948	1032
00478	1795	3421	2740
01070	1545	2802	2110
02040	703	1311	1002
03020	596	1088	824
65575	2208	3002	915
83565	1901	2269	1093
02575	2119	2922	1647
21585	2198	3293	2096
02050	1559	1422	1115
01565	2024	2032	1571

TABLA 4.3. TABLA DE REGISTROS DE TITANLUX Y BRUGUER

Fig. 4.1. Colores registrados.

Fijándose en los datos obtenidos para cada fabricante de pinturas, puede verse que los valores medidos en las pinturas de ”Bruguer” son, por lo general, mayores que los medidos en ”Titanlux”. Esto es debido a la reflexión de la luz, las medidas se realizan

iluminando el material con un led situado en el interior del sistema. Hay materiales que poseen una superficie de medida más reflectante que otra, esto se reproduce en el sistema desarrollado como un aumento en el nivel digital medido ya que, la luz del led es reflejada directamente al detector, recibiendo este, el color medido más esta luz reflejada [30].

Los colores de referencia de Bruguer han sido obtenidos mediante una revista de promoción, cuyas hojas poseen un acabado que las hace más reflectantes que el propio papel impreso normal, del cual se han conseguido los colores de referencia de Titanlux. Esto justifica el aumento de los valores dados por el sistema.

Por falta de tiempo y medios no se ha podido contrarrestar este efecto en el proyecto, en su lugar, se proponen varias medidas que, implementadas en un futuro, podrían eliminar este exceso de energía que recibe el detector, ofreciendo unos valores constantes para cualquier tipo de material.

- **Medir con ángulo:** Actualmente, las medidas se realizan de forma perpendicular, colocando el detector en posición vertical debajo del color que se desea medir. Realizando un estudio de la reflexión del color, es posible evitar el envío de energía reflejada al detector realizando la medida con cierto ángulo respecto al detector.
- **Utilizar un polarizador:** Colocando un polarizador entre el detector y el color medido se puede eliminar gran parte de la luz reflejada, recibiendo exclusivamente la energía emitida por el color [31].
- **Utilizar otro modelo de color:** Otra opción sería utilizar el modelo de color "HSL", el cual ofrece tres variables de valores (H, S, L), siendo una de ellas el matiz del mismo. Este valor no se ve afectado por la cantidad de luz reflejada en el color, permaneciendo constante ante distintos materiales, con diferentes niveles de reflectancia, pero con idénticos colores pintados en ellos.

4.3. Prueba 3 - Comparación entre colores de las diferentes paletas.

Finalmente, una vez se han registrado en la aplicación las paletas de colores, se procede a realizar las comparaciones entre ellos. Se han realizado las siguientes comparaciones: Entre los mismos colores registrados, usando colores no registrados y con objetos cotidianos. En todo momento los resultados han sido satisfactorios, ofreciendo unos colores muy próximos al comparado. A continuación, se muestran unas imágenes con los objetos y colores comparados donde observar estos resultados.

La primera prueba llevada a cabo es la comparación entre colores ya registrados en la aplicación. Es una prueba sencilla pero fundamental ya que, para su correcto funcionamiento, la aplicación debe ser capaz de reconocer los colores ya registrados.

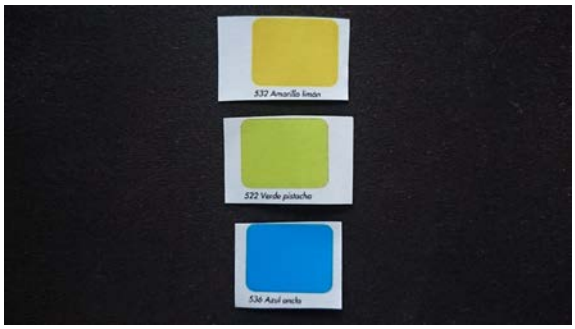


Fig. 4.2. Colores escogidos para la comparación.

Estos colores (figura 4.2) ya han sido registrados previamente en la aplicación. Se espera que el color recomendado por la aplicación, al comparar, sea mismo que se está midiendo en cada caso.

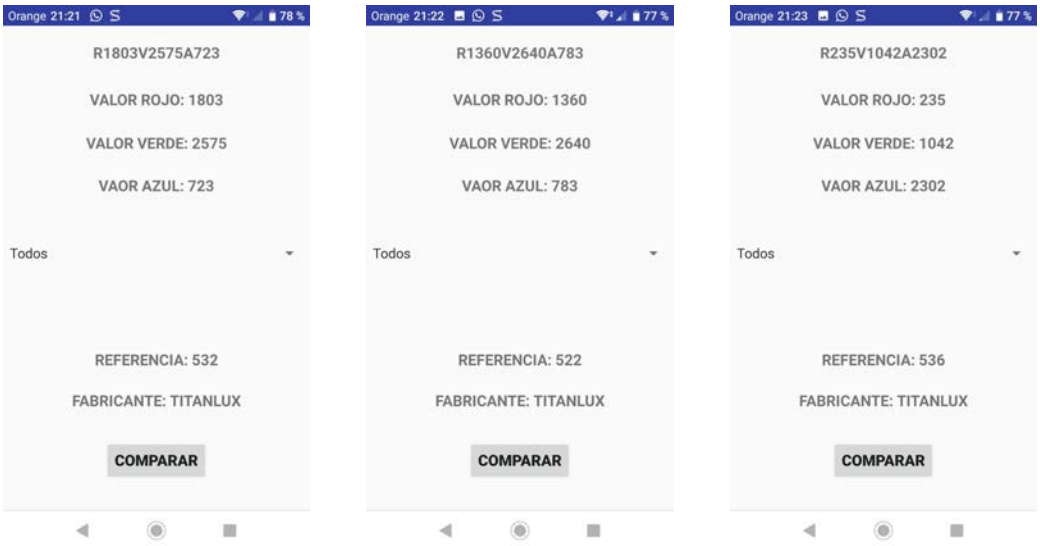


Fig. 4.3. Resultado de las comparaciones de los tres colores registrados.

Después de comprobar la comparación con colores ya registrados se ha utilizado un color de la paleta que, expresamente, se ha dejado sin registrar. La finalidad es que la aplicación muestre un color lo más parecido posible al comparar. En este caso la comparación se realizará dos veces, primero comparando entre todos los colores registrados y después comparando exclusivamente los colores de un fabricante.

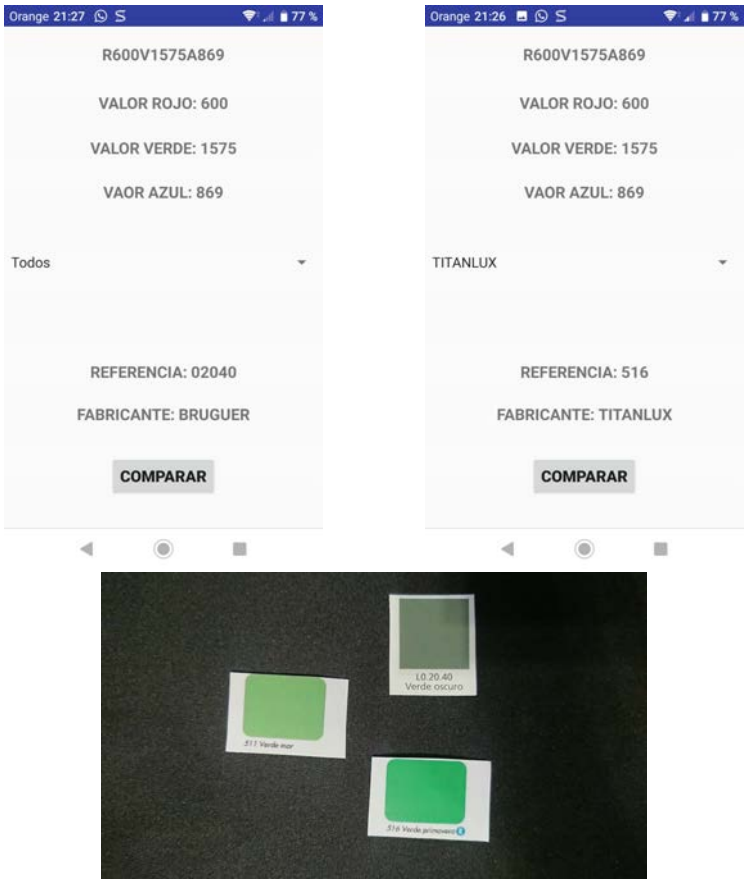


Fig. 4.4. Comparación de un color no registrado previamente.

En las pantallas de la aplicación, de la figura 4.4, puede verse que, en la primera pantalla la comparación se ha realizado utilizando todos los colores registrados, obteniendo como resultado color del fabricante Bruguer. En la segunda pantalla se ha escogido comparar exclusivamente con los colores del fabricante Titanlux. En la imagen inferior el color situado a la izquierda corresponde con el medido y comparado por la aplicación. Los colores de la derecha son los propuestos por la aplicación como los más parecidos.

4.4. Prueba 4 - Comparación entre objetos encontrados por casa.

En la última prueba llevada a cabo se han utilizado dos objetos encontrados por casa. Uno de ellos un cubo de rubik, más concretamente un ”Megaminx”, del cual se han comparado los colores de varias de sus caras. Otro, un paquete de post-itss ordinario, en este caso se ha comparado con los colores de Titanlux y Bruguer por separado. A continuación, se presentan imágenes de los resultados obtenidos. A la izquierda se encuentran los resultados de la comparación, a la derecha están las imágenes del objeto medido junto con el color obtenido por la aplicación como el más similar.

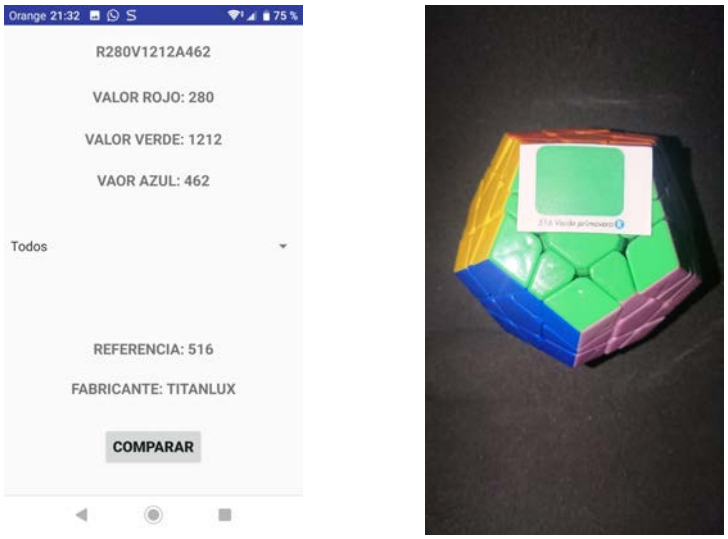


Fig. 4.5. Comparación cara verde.

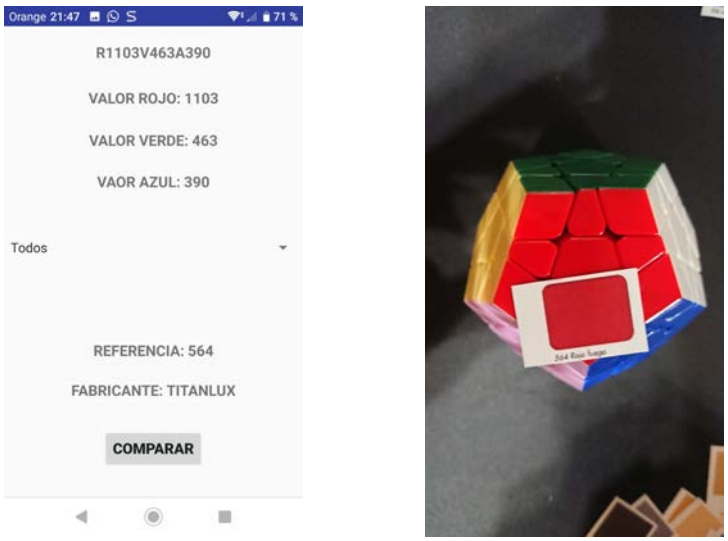


Fig. 4.6. Comparación cara roja.

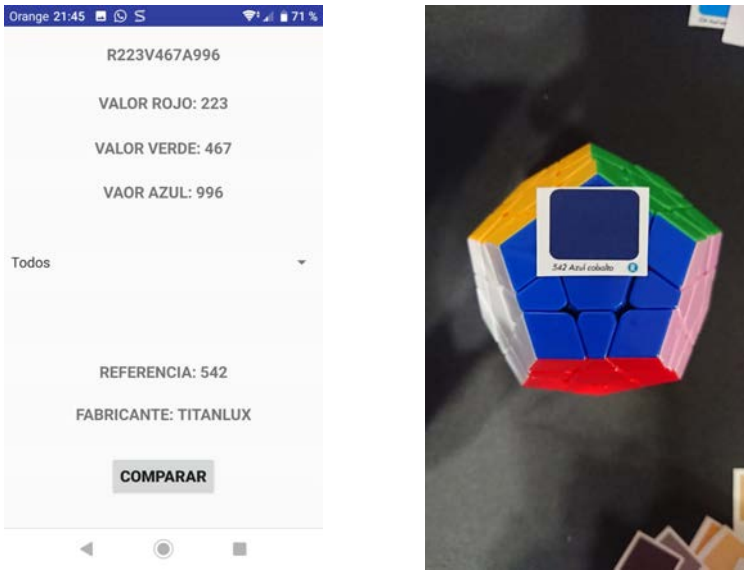


Fig. 4.7. Comparación cara azul.

Fig. 4.8. Comparación cara amarilla.

Como se puede ver, en estas cuatro figuras, los colores recomendados por la aplicación tienen un alto parecido a las caras medidas y comparadas del cubo.

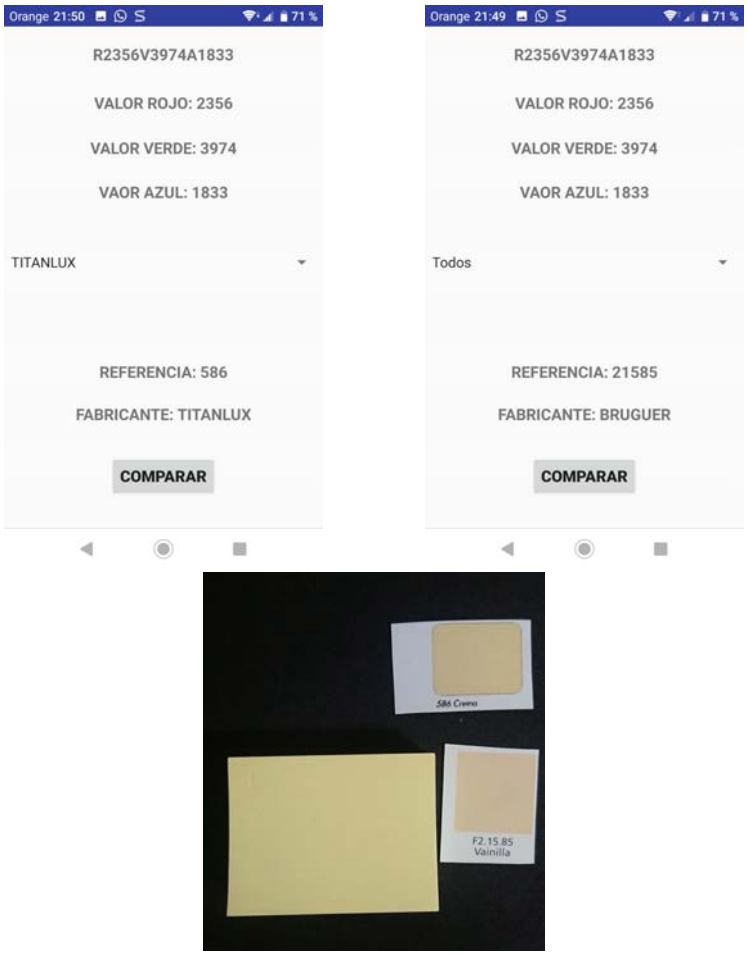


Fig. 4.9. Comparación post-its.

Para esta comparación, la aplicación indicaba que el color más parecido entre todos los registrados es el color de Bruguer. Con el fin de tener una segunda referencia se realizó una segunda comparación acotando los colores comparados a los del fabricante Titanlux. En la imagen pueden verse los colores de ambos fabricantes al lado del post-it medido, ambos son altamente parecidos.

4.5. Prueba 5 - Comparación bajo distinta iluminación ambiente.

Para terminar, se realizan varias comparaciones variando la iluminación de la habitación. Se pretende mostrar que la luz ambiente no afecta a las medidas que se realizan y, por lo tanto, tampoco a las comparaciones de estas.

Estos son los objetos que se ha decidido medir: Una de las caras de un cubo Rubik, un color de la gama de colores y la piel de un peluche rojo.

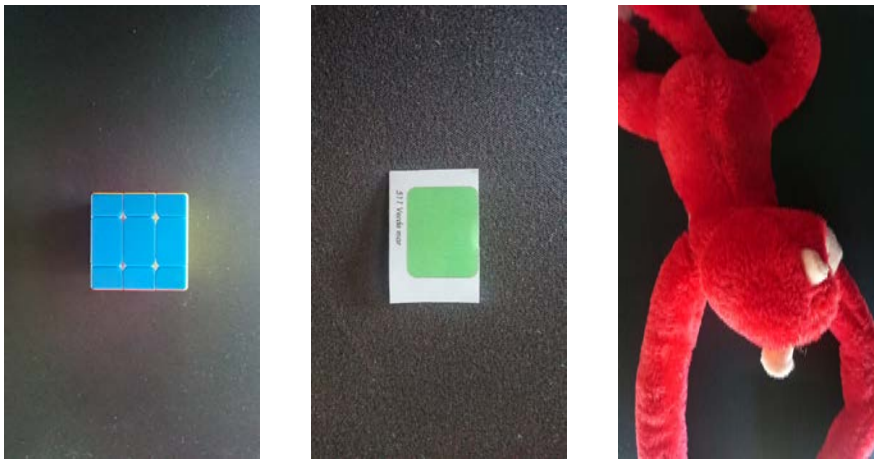


Fig. 4.10. Objetos medidos.

A continuación, se presentan las medidas realizadas con una fuerte iluminación en la habitación. Han sido tomadas a las 12:00 a.m. en una habitación con las persianas subidas donde el sol está impactando directamente.

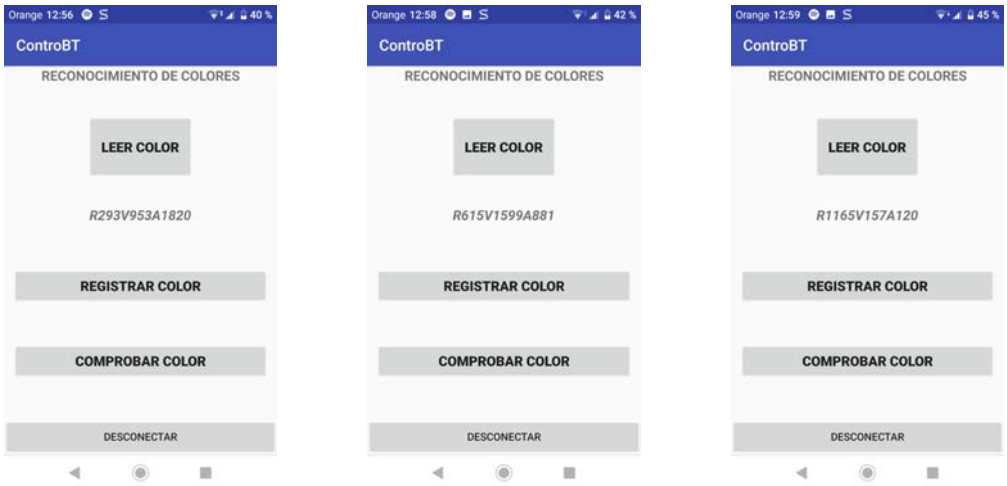


Fig. 4.11. Medidas con mucha iluminación.

Las siguientes medidas son las tomadas con baja iluminación. La habitación se ha dejado prácticamente a oscuras, cerrando del todo las persianas y apagando cualquier luz.

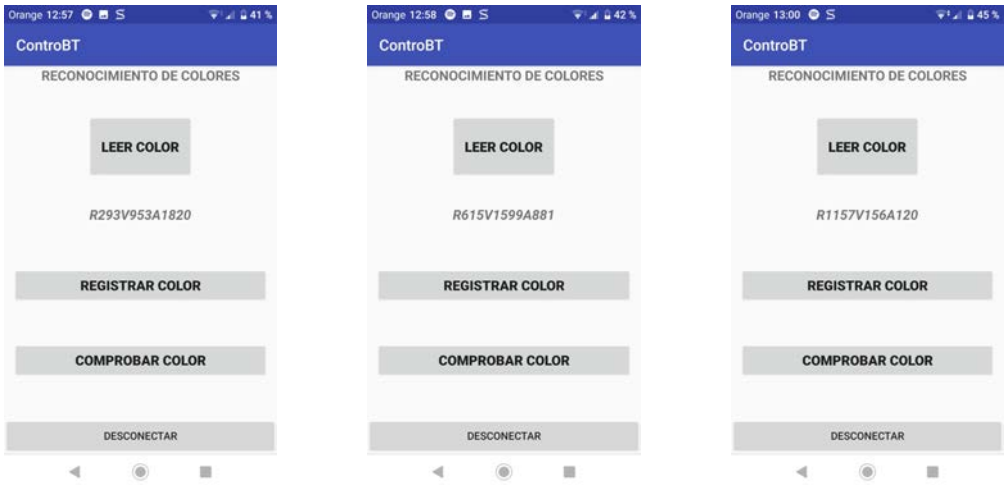


Fig. 4.12. Medidas con poca iluminación.

Como se puede observar en los resultados, la medida no varía cambiando la luz ambiente y, por lo tanto, la comparación realizada de cada uno de los colores tampoco variará.

4.6. Conclusiones acerca de los resultados obtenidos.

Tras ver los resultados obtenidos en cada una de las comparaciones realizadas se puede discernir que, esta aplicación, es apta para la utilización por cualquier usuario que quiera tener referencias de fabricantes de pintura. El sistema se ha encapsulado correctamente por lo que los usuarios podrán realizar medidas sin importan las condiciones ambientales.

5. PRESUPUESTO Y PLANIFICACION

5.1. PRESUPUESTO

El proyecto lo ha desarrollado una persona en solitario, posee un nivel profesional equivalente a un ingeniero junior. La duración del proyecto abarca unos 12 meses, en los que no se ha seguido una jornada de trabajo continua. El proyecto ha sido elaborado siguiendo la disponibilidad de tiempo del trabajador. Haciendo un recuento de días y horas trabajadas, el proyecto se ha elaborado tras 528 horas de trabajo.

El presupuesto del proyecto se divide en los siguientes apartados:

RECURSOS HUMANOS:

El sueldo medio de un ingeniero junior se ha establecido en 15 (€/h). Como solo ha trabajado una persona, un total de 528 horas, el coste de los recursos humanos asciende a: 15 [€] * 528 [€/h] = 7920 [€]

Se recoge en la siguiente tabla:

Componente	**Coste (€)**
Ingeniero junior	7920
Total recursos humanos	**7920**

TABLA 5.1. COSTES RECURSOS HUMANOS

MATERIALES:

Este apartado puede dividirse en dos tablas, la primera haciendo referencia al coste del hardware utilizado y otra al software y consumibles.

La tabla del hardware queda de la siguiente manera:

Componente	**Coste (€)**
Ordenador personal	1000
Procesador Arduino	20
Componente FTDI	6
Led blanco	0.10
Placa PCB	20
Equipo de soldaje	20
Caja para encapsulado	15
Herramientas (Taladro, lima, etc...)	30
Total hardware	**1111.1**

TABLA 5.2. COSTES HARDWARE

Respecto al software, como los IDE utilizados para la programación, el diseño de la placa y la realización de la memoria, ”Android Studio”, ”Arduino”, ”Kicad” y ”Latex” son de uso libre, es decir, gratuitos, por lo que no se añaden a los costes de software.

Componente	**Coste (€)**
Windows 10	35
Electricidad consumida	60
Material de oficina (papel, impresora, bolígrafos...)	20
Total software y consumibles	**115**

TABLA 5.3. COSTES SOFTWARE Y CONSUMIBLES

RESUMEN DEL PRESUPUESTO:

Apartado	**Coste (€)**
Recursos humanos	7920
Hardware	1111.1
Software y consumibles	115
Coste total	**9146.1**

TABLA 5.4. COSTES TOTALES

Se obtiene que, el coste total de proyecto asciende a 9146.1€.

5.2. PLANIFICACIÓN TEMPORAL

En lo referente a la planificación del sistema, se ha elaborado un diagrama de Gantt donde queda recogido el tiempo empleado en cada proceso realizado del proyecto. El diagrama se ha dividido en cuatro apartados generales correspondientes a cada parte del proyecto:

- Periodo de documentación: Tiempo en el que se busca información acerca de la tecnología que se pretende utilizar, así como estudios acerca de la teoría y el concepto que se va a explorar en el proyecto. Una vez se tienen los conceptos claros y se conoce la tecnología a usar, se puede elaborar una primera idea del sistema con la que comenzar el desarrollo.
- Desarrollo del hardware y software de Arduino: Este apartado corresponde al grueso del sistema. Hace referencia a la elaboración completa del sistema físico del proyecto. Abarca desde que se inician las conexiones entre componentes, hasta que se establece el sistema en un encapsulado preparado para ser utilizado, pasando también por los periodos de aprendizaje en los cuales se estudian las aplicaciones que se utilizan en la elaboración del sistema.

- Desarrollo de la aplicación en Android: Este apartado corresponde a la programación realizada para la elaboración de la aplicación utilizada en el proyecto. Además, abarca el periodo de tiempo destinado a recoger y estudiar información acerca del software utilizado para la programación.
- Pruebas finales del sistema completo: La fase final de pruebas tiene lugar una vez se han finalizado las fases anteriores, estando el sistema completado y funcionando eficientemente.
- Redacción del documento: La memoria se comenzó a elaborar antes de haber finalizado por completo el sistema. Una vez se han realizado las pruebas finales, y los resultados han sido satisfactorios, la memoria ha podido ser terminada.

A continuación, se puede ver el diagrama de Gantt que recoge el periodo de tiempo que se ha dedicado a cada apartado.

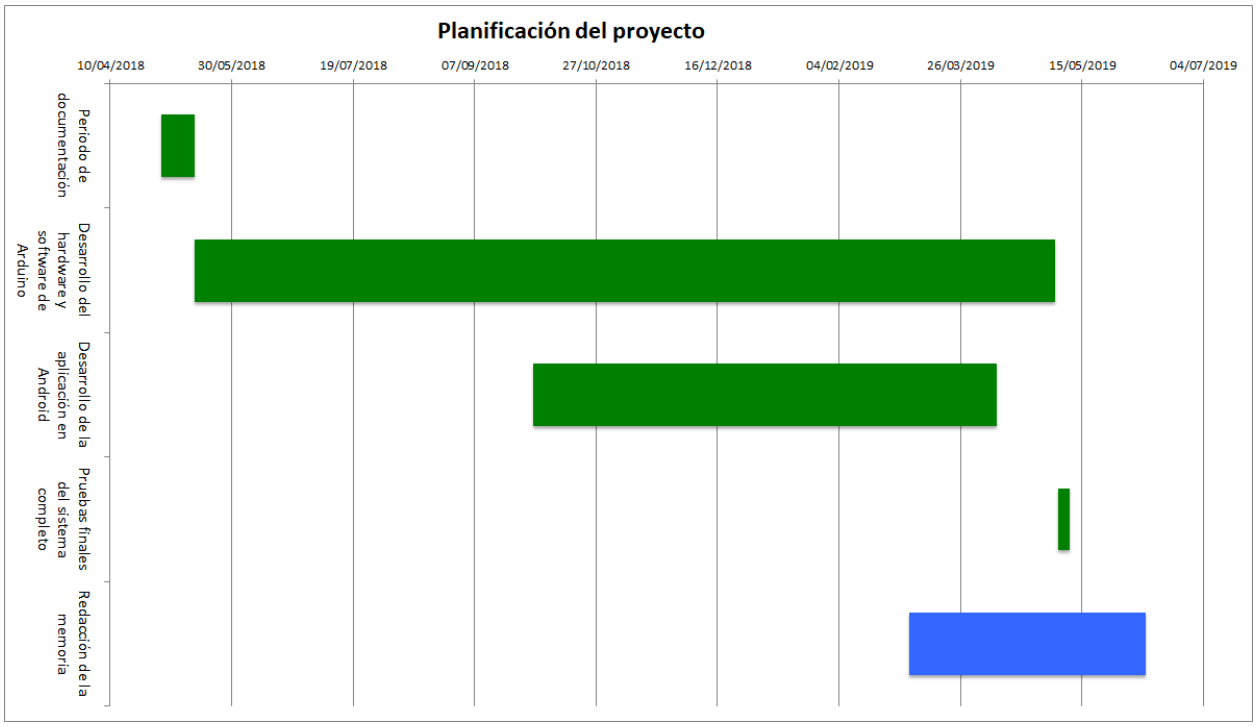


Fig. 5.1. Diagrama de Gantt.

6. CONCLUSIONES Y TRABAJO FUTURO

Como conclusión, se exponen una serie de conclusiones con la finalidad de servir como reflexión final al trabajo realizado, presentando la resolución de los objetivos cumplidos y concluyendo con un listado de los posibles trabajos futuros para el sistema.

Se ha conseguido cumplir el objetivo principal del proyecto. Gracias a los medios utilizados y el desarrollo establecido, se ha construido una base de datos capaz de almacenar medidas realizadas a través de un sistema basado en Arduino. Esta base de datos almacena gran cantidad de colores de distintos fabricantes, de tal forma que está preparada para que el usuario sea capaz de realizar sus propias comparaciones desde un inicio. También incluye la posibilidad de registrar sus propios colores, en caso de que se quiera ampliar la base de datos.

Gracias al estudio de la física relacionada con la colorimetría, ha sido posible entender y trabajar directamente con la luz en todas sus formas, obteniendo valores medibles para cada color visible y pudiendo entender y trabajar con ellos.

El software realizado ha cumplido con los objetivos previstos ya que, es capaz de diferenciar, con buena sensibilidad, los diferentes colores que se presenten, obteniendo unos errores mínimos cuando un mismo color es medido repetidas veces. Además, Se ha conseguido diseñar una interfaz sencilla e intuitiva para los usuarios.

Además, el trabajador ha conseguido obtener conocimientos acerca de la colorimetría, la realización de aplicaciones donde se trabaja con bluetooth y programación en Android, el uso de Arduino y su programación, la fabricación de placas PCBs y su desarrollo. Al comienzo del proyecto todos estos conocimientos resultaban desconocidos para el trabajador.

Como el tiempo y los recursos han sido limitados, no ha sido posible llevar a cabo todas las ideas de mejora que han ido surgiendo conforme se realizaba el proyecto, ya sean para mejorar su funcionamiento, o para ampliar los objetivos marcados. Se elabora a continuación una lista de todas las mejoras que podrían llevarse a cabo como un trabajo futuro con la finalidad de obtener los mejores resultados con el sistema.

- Clasificación de los colores según el tipo de soporte de impresión. Como se ha explicado durante la memoria, la textura de los diferentes objetos importa a la hora de medir su color. Un objeto con una textura más reflectante que otro, aún con el mismo color, resulta en un nivel de intensidad reflejado mayor. El trabajo futuro propuesto no es solo poder corregir este fenómeno, con el uso de polarizadores o usando otro modelo de color para representar los datos, sino, también ser capaces de diferenciar el nivel de reflectancia que posee el objeto. Esto mismo puede conseguirse al utilizar otro modelo de color para representar los datos, un modelo donde un valor indique el nivel de luminosidad recibido. Como en el sistema el ni-

vel de luminosidad es constante, generado por el led, resultaría sencillo comprobar la cantidad de luz extra reflejada por el objeto.

- Una forma de mejorar la aplicación será colorear la pantalla con el matiz del color medido para cada adquisición. De esa forma resultará más visual si la adquisición realizada es correcta.

- Como futuro trabajo se pensó en la implementación de audio en la aplicación, de tal manera que la aplicación pueda ser capaz de nombrar el color que se ha medido. Para esta función no es necesaria la comparación de colores con los registrados por los fabricantes. Esta función resultaría muy útil en sí misma ya que, ayudaría a las personas con problemas de visión en su día a día a reconocer los colores de su alrededor si les fuese necesario.

- Reducir el tamaño del sistema. El sistema final tiene un tamaño mayor de lo esperado y, aunque sigue siendo fácil de transportar, mejorando el diseño y con un mayor coste económico, será posible reducir su tamaño aún más.

BIBLIOGRAFÍA

[1] W. Castañ *et al.*, *Color*. Universidad de Caldas, 2005.

[2] G. Saravanan, G. Yamuna, and S. Nandhini, "Real time implementation of rgb to hsv/hsi/hsl and its reverse color space models," in *2016 International Conference on Communication and Signal Processing (ICCSP)*, pp. 0462–0466, IEEE, 2016.

[3] J. Itten, *Arte del color: aproximación subjetiva y descripción objetiva del arte*. Bouret, 1975.

[4] V. Autores, "Espectro visible," 2019. Wikipedia. https://es.wikipedia.org /wiki/Espectro_visible. (acceso: 10-03-2019).

[5] C. Urtubia Vicario, *Neurobiología de la visión*, vol. 51. Universitat Politècnica de Catalunya. Iniciativa Digital Politècnica, 2004.

[6] V. Autores, "Prisma," 2019. Spaceplace. https://spaceplace.nasa.gov/review/magic-windows/vis.sp.html. (acceso: 10-03-2019).

[7] V. Autores, "Colores primarios," 2019. Quora.https://es.quora.com/C %C3 %B3mo-se-nombraron-los-colores-primarios-Una-persna-asign %C3 %B3-palabras-para-representar-cada-uno-de-los-colores (acceso: 10-03-2019).

[8] K. Sayanagi, "Method and apparatus for reproducing a color image using additive and subtractive primary colors," Sept. 30 1986. US Patent 4,614,967.

[9] R. Mendoza, "¿Cómo saber si un televisor es verdaderamente 4k? (videos y fotos)," 2019. https://diariocorreo.pe/tecnologia/como-saber-si-un-televisor-es-verdaderamente-4k-videos-y-fotos-827015/ (acceso: 10-03-2019).

[10] V. Autores, "Colores aditivos," 2019. Wikimedia. https://commons.wikimedia. org/wiki/File:AdditiveColorSynthesis_RGBpositives.jpg (acceso: 10-03-2019).

[11] G. Wyszecki and W. S. Stiles, *Color science*, vol. 8. Wiley New York, 1982.

[12] D. Litwiller, "Ccd vs. cmos," *Photonics Spectra*, vol. 35, no. 1, pp. 154–158, 2001.

[13] HAMAMATSU PHOTONICS K.K., *Color sensor*, may 2018. https://automotive. hamamatsu.com/resources/pdf/ssd/s9706_kpic1060e.pdf (acceso: 10-03-2019).

[14] S. L. Tewinkle, "Image sensor with integration time compensation," July 3 2012. US Patent 8,212,197.

[15] V. Autores, "Escalas cromáticas, acromáticas y gamas de colores." Fotonostra.com. https://www.fotonostra.com/grafico/escalascolores.htm (acceso: 15-03-2019).

[16] V. Autores, “Escalas cromáticas, acromáticas y gamas de colores,” 2019. Fotonostra. https://www.fotonostra.com/grafico/escalascolores.htm (acceso: 10-03-2019).

[17] V. Autores, “Modelo de color hsv,” 2019. Wikipedia. https://es.wikipedia.org/wiki/Modelo_de_color_HSV/media/File:Cono_de_la_coloraci %C3 %B3n_HSV.png (acceso: 10-03-2019).

[18] V. Autores, “Scribble,” 2019. La brujula Verde. https://www.labrujulaverde.com/2014/06/scribble-un-boligrafo-para-dibujar-con-cualquier-color-que-exista-en-el-mundo (acceso: 10-03-2019).

[19] V. Autores, “Cronzy,” 2019. Ideakreativas. https://ideakreativa.net/pluma-cronzy-mas-de-16-millones-de-colores-en-su-bolsillo/ (acceso: 10-03-2019).

[20] V. Autores, “Arduino mini 05,” 2019. Store.arduino.cc. https://store.arduino.cc/arduino-mini-05 (acceso: 10-03-2019).

[21] V. Autores, “Modulo bluetooth,” 2019. Tienda bricogeek. https://tienda.bricogeek.com/modulos-bluetooth/800-modulo-bluetooth-hc-05.html (acceso: 10-03-2019).

[22] V. Autores, “Ftdi ft232rl conversor,” 2019. Electronicastore. https://electronicastore.net/producto/modulo-ftdi-ft232rl-convertidor-usb-ttl/ (acceso: 10-03-2019).

[23] V. Autores, “Led blanco,” 2019. Amazon. https://www.amazon.es/50X-Diodo-Blanco-alta-luminosidad/dp/B07CZYQZBT/ref=sr_1_12?keywords=diodo+led+blancoqid=1559833543s= gatewaysr=8-12 (acceso: 7-04-2019).

[24] V. Autores, “Kicad eda.” Kicad-pcb.org. http://www.kicad-pcb.org/ (acceso: 10-03-2019).

[25] M. P. Aparicio, *Diseño y desarrollo de circuitos impresos con KiCad*. Rc Libros, 2010.

[26] J. M. R. Gutiérrez, “Manual de programación arduino,” *Transl.: BW Evans et al., Arduino Notebook: A Beginner’s Reference*.

[27] J. T. Gironés, *El gran libro de Android*. Marcombo, 2012.

[28] V. Autores, “Arduino - software.” Arduino.cc. https://www.arduino.c c/en/Main/-Software (acceso: 10-03-2019).

[29] V. Autores, “Android studio and sdk tools.” Android Developers. https://developer.android.com/studio (acceso: 5-03-2019).

[30] J. L. Caivano and M. A. López, *Color: ciencia, artes, proyecto y enseñanza. Argencolor 2004*. Editorial Nobuko, 2000.

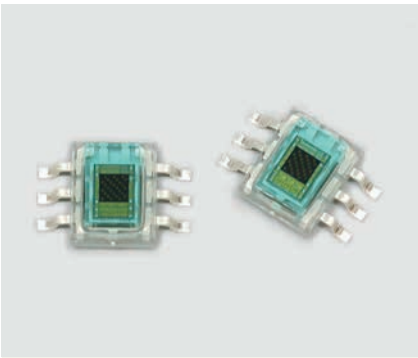
[31] J. R. García and J. M. V. Rovira, *Fundamentos de óptica ondulatoria*. Universidad de Oviedo, 1998.

ANEXO

Anexo A: Datasheet S9706

HAMAMATSU
PHOTON IS OUR BUSINESS

Color sensor



S9706

12-bit digital output

The S9706 is a digital color sensor sensitive to red (λ=615 nm), green (λ=540 nm) and blue (λ=465 nm) regions of the spectrum. Detected signals are serially output as 12-bit digital data. Built-in three 12-bit registers allow simultaneous measurement of RGB three colors. Sensitivity level is adjustable in two steps to cover a wide photometric range.

Features

- **12-bit digital output**
- **Simultaneous measurement of RGB three colors**
- **2-step sensitivity switching (sensitivity ratio of 1 : 9)**
- **Low voltage (3.3 V) operation**
- **CMOS monolithic photo IC**
- **No external components required**

Applications

- **Display color adjustment**
- **Various applications involving color detection**

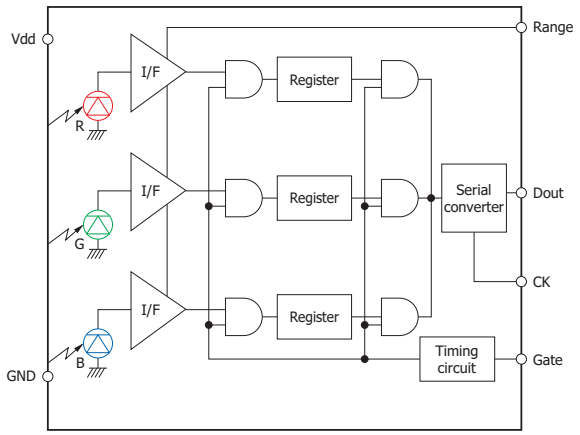
Feature 01 12-bit digital output

Light signals detected by the photodiode are amplified and converted into 12-bit digital signals. An amplifier is also formed for each of the RGB photodiode elements arrayed in the mosaic pattern, allowing simultaneous accurate measurement of the RGB components of incident light.

Feature 02 Simultaneous measurement of RGB three colors

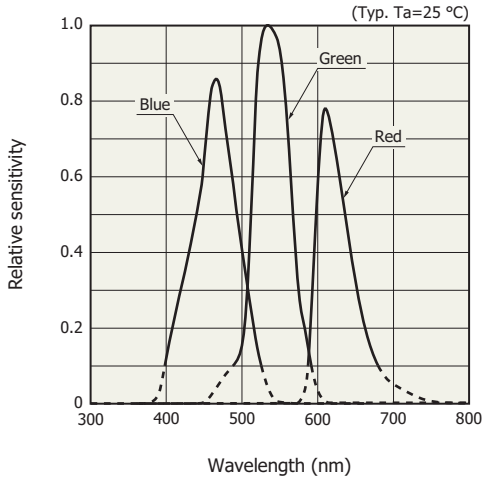
The photodiode consists of 9 × 9 elements arrayed in a mosaic pattern. Each element has an on-chip filter that it sensitive to one color of light, either red (λp=615 nm), green (λp=540 nm) or blue (λp=465 nm).

Block diagram



KPICC0110EB

Spectral response



KPICB0089EA

This product does not support lead-free soldering. Solder it by hand.

Feature 03 2-step sensitivity switching

To enable measurement over a wide range of illuminance, the photodiode sensitivity can be selected from two setting modes (high sensitivity mode and low sensitivity mode). The photodiode photosensitive area used to detect light differs depending on which sensitivity mode is selected (high sensitivity mode: 9 × 9 elements, low sensitivity mode: 3 × 3 elements in center).

Sensitivity setting

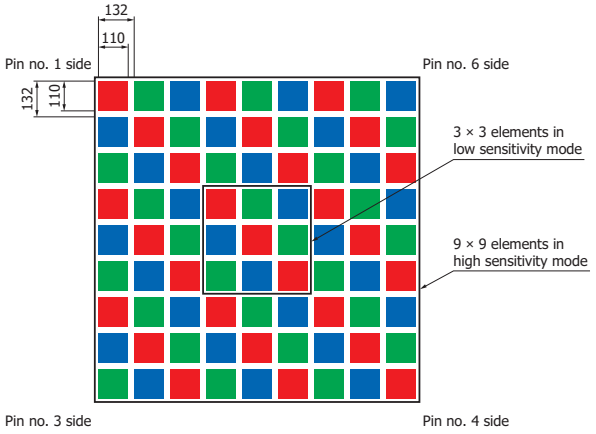
Range	Mode	Effective photosensitive area*
High	High sensitivity	9 × 9 elements
Low	Low sensitivity	3 × 3 elements

* The photosensitive area of S9706 consists of 9 × 9 elements in a mosaic pattern.

The effective photosensitive area changes depending on which sensitivity mode is used, "high" or "low", as explained below.

· High sensitivity mode: 9 × 9 elements
· Low sensitivity mode: 3 × 3 elements in center

Details of photosensitive area (unit: µm)



Note: Spaceing between elements is light-shielded.

KPICC0124EB

Absolute maximum ratings

Parameter	Symbol	Condition	Value	Unit
Supply voltage	Vdd	Ta=25 °C	-0.3 to 6	V
Load current	Io	Ta=25 °C	±10	mA
Power dissipation	P	Ta=25 °C	100	mW
Operating temperature	Topr		-20 to +85	°C
Storage temperature	Tstg		-20 to +85	°C

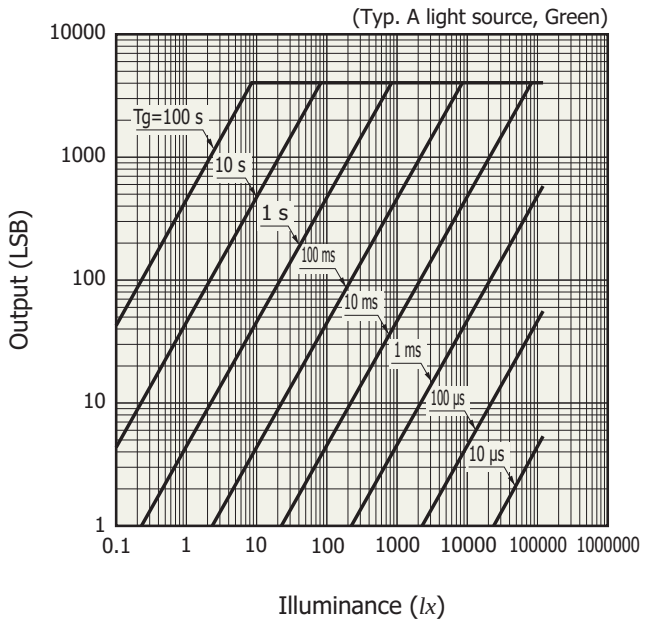
Note: Exceeding the absolute maximum ratings even momentarily may cause a drop in product quality. Always be sure to use the product within the absolute maximum ratings.

Electrical and optical characteristics (Ta=25 °C, Vdd=5 V, Tg=100 ms, A light source, unless otherwise noted)

Parameter	Symbol	Condition	Min.	Typ.	Max.	Unit
Photosensitive area size	-	All elements (9 × 9 elements)	-	1.2 × 1.2	-	mm
Effective photosensitive area	-	Per 1 color, High range	-	0.32	-	mm^2
Spectral response range	λ	Blue	-	400 to 540	-	nm
		Green	-	480 to 600	-	
		Red	-	590 to 720	-	
Peak sensitivity wavelength	λp	Blue	-	465	-	nm
		Green	-	540	-	
		Red	-	615	-	
Supply voltage	Vdd		3.0	-	5.5	V
Current consumption	Idd	Dark state, no load	-	5	10	mA
Photosensitivity	Sbl	Blue, Low range	0.15	0.21	0.27	LSB/*lx*
	Sgl	Green, Low range	0.32	0.45	0.59	
	Srl	Red, Low range	0.45	0.64	0.83	
	Sbh	Blue, High range	1.3	1.9	2.5	
	Sgh	Green, High range	2.8	4.1	5.4	
	Srh	Red, High range	4.0	5.8	7.6	
Incident light power (Conversion value in A light source)	Ibl	Blue, Low range	-	-	240	k*lx*
	Igl	Green, Low range	-	-	110	
	Irl	Red, Low range	-	-	78	
	Ibh	Blue, High range	-	-	26	
	Igh	Green, High range	-	-	12	
	Irh	Red, High range	-	-	8.6	
Dark output	Dark	Tg=0.5 s	-	-	1	LSB
Input high level	Vih		Vdd × 0.82	-	-	V
Input low level	Vil		-	-	Vdd × 0.18	V
High level output voltage	Voh	Ioh=-0.5 mA	4.5	-	-	V
Low level output voltage	Vol	Iol=0.5 mA	-	-	0.5	V
Integration time	Tg		Refer to “Output vs. illuminance”			-
Hold time	t1		4	-	-	µs
	t2		3	-	-	µs
	t3		3	-	-	µs
	t4		2000	-	-	µs
	t5		3	-	-	µs
Readout clock period	tck		500	-	-	ns
Readout pulse width (positive)	tw		200	-	-	ns
Readout pulse width (negative)	tck-tw		200	-	-	ns

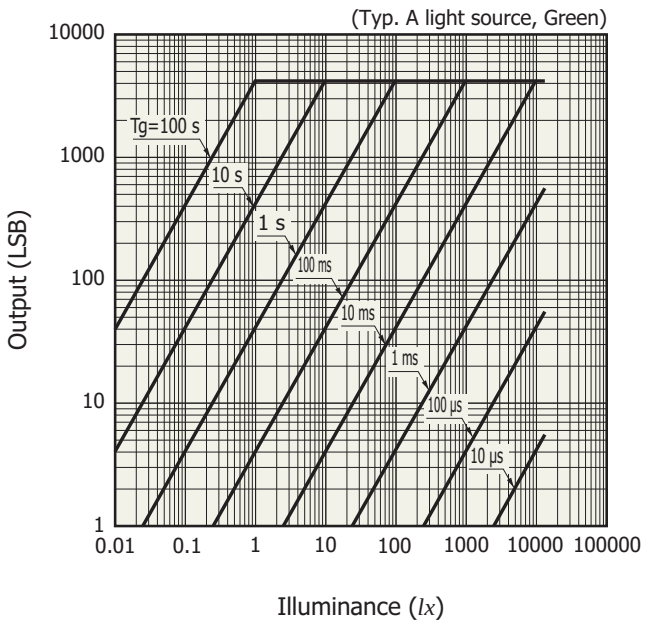
Output vs. illuminance

Low range



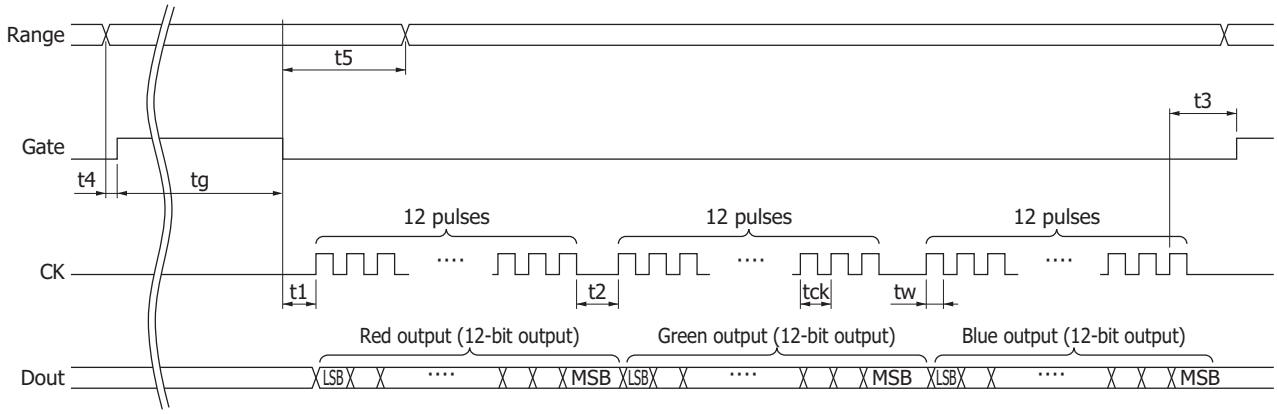
KPICB0099EA

High range



KPICB0100EA

Timing chart

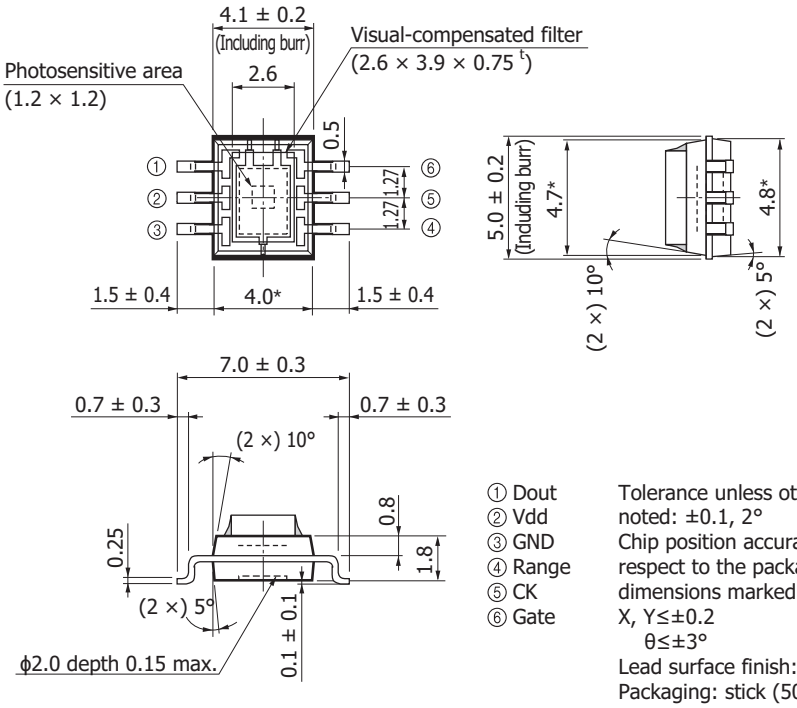


Operating sequence
(1) Set the Gate terminal and CK terminal to "Low".
(2) Select the desired sensitivity with the Range terminal.
(3) Set the Gate terminal from "Low" to "High", to start integrating the light intensity.
(4) After the desired integration time (tg) has passed, set the Gate terminal from "High" to "Low" to end the light intensity integration.
(5) Measurement data is output from the Dout terminal by inputting 36 CK pulses to the CK terminal.

Note 1: A total of 36 CK pulses are required to read out 3-color measurement data. Red data is output by the first 12 pulses, green data by the next 12 pulses, and blue data by the last 12 pulses. Measurement data is output from the LSB side.
Note 2: Measurement data changes at the CK pulse rising edge.
Note 3: Do not switch the Range terminal during integration time (tg).

KPICC0115EB

Dimensional outline (unit: mm)



① Dout
② Vdd
③ GND
④ Range
⑤ CK
⑥ Gate

Tolerance unless otherwise noted: ±0.1, 2°
Chip position accuracy with respect to the package dimensions marked *
X, Y≤±0.2
θ≤±3°
Lead surface finish: silver plating
Packaging: stick (50 pcs/stick)

KPICA0060EE

Note: If excessive vibration is continuously applied to the glass filter, there is a risk that the filter may come off, so secure the glass filter with a holder.

Line-up of RGB color sensors

Type no.	Type	Photosensitive area (mm)	Package (mm)	Peak sensitivity wavelength (nm)	Photosensitivity						Photo
S9032-02	Photodiode	 ϕ2.0	4 × 4.8 × 1.8[t] 6 pin (filter 0.75[t])	B 460 G 540 R 620	B 0.18 (A/W) [λ=460 nm] G 0.23 (A/W) [λ=540 nm] R 0.16 (A/W) [λ=620 nm]						
S9702	Photodiode	 1.0 × 1.0	3 × 4 × 1.3[t] 4 pin (filter 0.75[t])	B 460 G 540 R 620	B 0.18 (A/W) [λ=460 nm] G 0.23 (A/W) [λ=540 nm] R 0.16 (A/W) [λ=620 nm]						
S10917-35GT	Photodiode	 1.0 × 1.0	3 × 1.6 × 1.0[t] COB (on-chip filter)	B 460 G 540 R 620	B 0.2 (A/W) [λ=460 nm] G 0.23 (A/W) [λ=540 nm] R 0.17 (A/W) [λ=620 nm]						
S10942-01CT	Photodiode	 1.0 × 1.0	3 × 1.6 × 1.0[t] COB (on-chip filter)	*	B 0.21 (A/W) [λ=460 nm] G 0.25 (A/W) [λ=540 nm] R 0.45 (A/W) [λ=640 nm]						
S9706	Digital photo IC	 1.2 × 1.2	4 × 4.8 × 1.8[t] 6 pin (filter 0.75[t])	B 465 G 540 R 615	Low B 0.21 (LSB/lx) G 0.45 (LSB/lx) R 0.64 (LSB/lx)		High B 1.9 (LSB/lx) G 4.1 (LSB/lx) R 5.8 (LSB/lx)				
S11012-01CR	Digital photo IC	 1.2 × 1.2	3.43 × 3.8 × 1.6[t] COB (on-chip filter)	*	Low B 0.3 (LSB/lx) G 0.6 (LSB/lx) R 1.4 (LSB/lx)		High B 2.6 (LSB/lx) G 5.3 (LSB/lx) R 12.9 (LSB/lx)				
S11059-02DT /-03DS	I²C compatible color sensor	 0.56 × 1.22	3 × 4.2 ×1.3[t] 10 pin (on-chip filter)	B 460 G 530 R 615 IR 855	Low B 4.4 (count/lx) G 8.3 (count/lx) R 11.2 (count/lx) IR 3.0 (count/lx)		High B 44.8 (count/lx) G 85.0 (count/lx) R 117.0 (count/lx) IR 30.0 (count/lx)				

* Refer to the spectral response of each product's datasheet.

Related information

www.hamamatsu.com/sp/ssd/doc_en.html

■ Precautions

- Disclaimer
- Metal, ceramic, plastic package products
- Surface mount type products

HAMAMATSU www.hamamatsu.com

HAMAMATSU PHOTONICS K.K., Solid State Division
1126-1 Ichino-cho, Higashi-ku, Hamamatsu City, 435-8558 Japan, Telephone: (81) 53-434-3311, Fax: (81) 53-434-5184
U.S.A.: Hamamatsu Corporation: 360 Foothill Road, Bridgewater, N.J. 08807, U.S.A., Telephone: (1) 908-231-0960, Fax: (1) 908-231-1218
Germany: Hamamatsu Photonics Deutschland GmbH: Arzbergerstr. 10, D-82211 Herrsching am Ammersee, Germany, Telephone: (49) 8152-375-0, Fax: (49) 8152-265-8
France: Hamamatsu Photonics France S.A.R.L.: 19, Rue du Saule Trapu, Parc du Moulin de Massy, 91882 Massy Cedex, France, Telephone: 33-(1) 69 53 71 00, Fax: 33-(1) 69 53 71 10
United Kingdom: Hamamatsu Photonics UK Limited: 2 Howard Court, 10 Tewin Road, Welwyn Garden City, Hertfordshire AL7 1BW, United Kingdom, Telephone: (44) 1707-294888, Fax: (44) 1707-325777
North Europe: Hamamatsu Photonics Norden AB: Torshamnsgatan 35 16440 Kista, Sweden, Telephone: (46) 8-509-031-00, Fax: (46) 8-509-031-01
Italy: Hamamatsu Photonics Italia S.r.l.: Strada della Moia, 1 int. 6, 20020 Arese (Milano), Italy, Telephone: (39) 02-93581733, Fax: (39) 02-93581741
China: Hamamatsu Photonics (China) Co., Ltd.: B1201, Jiaming Center, No.27 Dongsanhuan Beilu, Chaoyang District, Beijing 100020, China, Telephone: (86) 10-6586-6006, Fax: (86) 10-6586-2866

Cat. No. KPIC1060E14 Feb. 2016 DN

Anexo B: Código de Arduino

```
#include <S9706.h>
#include <SoftwareSerial.h>

SoftwareSerial BTSerial(8,7); //RX,TX

const int dataPin = 10; // No.1 Dout connected to digital pin 10
const int rangePin = 4; // No.4 Range connected to digital pin 4
const int clockPin = 5; // No.5 CK connected to digital pin 5 const
int gatePin = 6;  // No.6 Gate connected to digital pin 6 S9706
colorSensor(dataPin, rangePin, clockPin, gatePin);

void setup() {
Serial.begin(9600
);
pinMode(LED_B
UILTIN,
OUTPUT);
colorSensor.begin
();

}

void loop() {

 BTSerial.begin(9600);
 Serial.print("Pulse para iniciar lectura del color");

// BTSerial.print("Pulse para iniciar lectura del color: ");
// BTSerial.print("#");

 char x = 'a';
while(x!='1'){

  x = BTSerial.read();

}
Serial.print("Midiendo Color: ");
//BTSerial.println(" ");
//BTSerial.print("Midiendo Color: ");
//BTSerial.println("");
 // update RGB
```

```
 //
 // S9706::update(mode)
 //        mode:                specify S9706_MODE_HIGH or S9706_MODE_LOW
colorSensor.update(S9706_MODE_HIGH);

 //  S9706::setcolor(durationMs, durationUs = 0);
 //   durationMs:   specify sensor integration process time (milliseconds)  //   durationUs:
specify    sensor    integration    process    time    (microseconds)    or    not
digitalWrite(LED_BUILTIN, HIGH);   colorSensor.setcolor(950);
 digitalWrite(LED_BUILTIN, LOW);

 // get updated RGB (value min: 0, max:
4095)    int  red  =  colorSensor.getRed();
int green = colorSensor.getGreen();   int
blue = colorSensor.getBlue();

 // print line "red,geen,blue"
 Serial.print(red);
 Serial.print(",");
 Serial.print(green);
 Serial.print(",");
 Serial.print(blue);
 Serial.println("");

 //if(Serial.available()){

  BTSerial.print("R");
  BTSerial.print(red);

  BTSerial.print("V");
  BTSerial.print(green);

  BTSerial.print("A");
  BTSerial.print(blue);
  BTSerial.print("#");
 //BTSerial.println("");



 delay(5000);}
```

Anexo C: Código de Android

```
 1 package com.estudiantes.controbt;
 2
 3 import android.content.ContentValues;
 4 import android.content.Intent;
 5 import android.database.sqlite.SQLiteDatabase;
 6 import android.os.Bundle;
 7 import android.support.v7.app.AppCompatActivity;
 8 import android.view.View;
 9 import android.widget.Button;
10 import android.widget.EditText;
11 import android.widget.TextView;
12 import android.widget.Toast;
13
14 import com.estudiantes.controbt.utilidades.Utilidades;
15
16 public class prueba extends AppCompatActivity {
17
18
19     Button IdRegistro;
20     TextView IdDato, Idtexto, IdR,IdV,IdA;
21     EditText IdFabricante, IdReferencia;
22     int Drojo,Dverde,Dazul;
23
24     int g,b;
25
26
27
28
29
30     @Override
31     protected void onCreate(Bundle savedInstanceState) {
32         super.onCreate(savedInstanceState);
33         setContentView(R.layout.activity_prueba);
34
35
36
37         IdDato = (TextView) findViewById(R.id.Iddato);
38         IdR = (TextView) findViewById(R.id.Idr);
39         IdV = (TextView) findViewById(R.id.Idv);
40         IdA = (TextView) findViewById(R.id.Ida);
41         IdFabricante = (EditText) findViewById(R.id.
   Idfabricante);
42         IdReferencia = (EditText) findViewById(R.id.
   Idreferencia);
43         IdRegistro = (Button) findViewById(R.id.Idregistro
```

```
43 );
44
45
46         //Recibo lo enviado desde Activity_User
47         Intent miIntent = getIntent();
48         Bundle extras = miIntent.getExtras();
49
50         String dato_color = extras.getString("COLOR");
51         IdDato.setText(dato_color);
52         for (int i = 0; i < dato_color.length(); i++){
53            char x=dato_color.charAt(i);
54             if(x == 'V') { g = i; }
55             if(x == 'A') { b = i; }
56         }
57         String rojo = dato_color.substring(1,g);
58         String verde = dato_color.substring(g+1,b);
59         String azul = dato_color.substring(b+1,dato_color.
   length());
60
61         IdR.setText("VALOR ROJO: " + rojo);
62         IdA.setText("VAOR AZUL: " + azul);
63         IdV.setText("VALOR VERDE: " + verde);
64
65         Drojo = Integer.parseInt(rojo);
66         Dverde = Integer.parseInt(verde);
67         Dazul = Integer.parseInt(azul);
68
69
70
71
72
73         IdRegistro.setOnClickListener(new View.
   OnClickListener(){
74             @Override
75             public void onClick(View v)
76             {
77                 //Cuando le de al oton "Registrar" Lo que
   haya escrito en fabricante y referencia se guarda en
   variables junto a los colores y se registran en la base de
    datos.
78                 registrarColores();
79                //Registro en la base de datos los valores
    "ROJO" "VERDE" "AZUL" "REFERENCIA" "FABRICANTE"
80             }
81         });
```

```
82
83      }
84
85      public void onClick(View view){
86          registrarColores();
87      }
88
89      private void registrarColores() {
90          ConexionSQLiteHelper conn=new
    ConexionSQLiteHelper(this,"bd_colores",null,1);
91
92          SQLiteDatabase db=conn.getWritableDatabase();
93          ContentValues values = new ContentValues();
94          values.put(Utilidades.CAMPO_ROJO,Drojo);
95          values.put(Utilidades.CAMPO_VERDE,Dverde);
96          values.put(Utilidades.CAMPO_AZUL,Dazul);
97          values.put(Utilidades.CAMPO_REFERENCIA,
    IdReferencia.getText().toString());
98          values.put(Utilidades.CAMPO_FABRICANTE,
    IdFabricante.getText().toString());
99
100         Long idResultante=db.insert(Utilidades.
    TABLA_COLORES,Utilidades.CAMPO_REFERENCIA,values);
101
102         Toast.makeText(getApplicationContext(),"Id
    Registro: " + idResultante, Toast.LENGTH_SHORT).show();
103
104         db.close();
105
106     }
107
108
109
110
111 }
112
113
```

```java
package com.estudiantes.controbt;

import android.content.Intent;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.os.Bundle;
import android.support.v7.app.AppCompatActivity;
import android.view.View;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.Spinner;
import android.widget.TextView;

import com.estudiantes.controbt.entidades.Color;
import com.estudiantes.controbt.utilidades.Utilidades;

import java.util.ArrayList;
import java.util.concurrent.CountDownLatch;

public class Comparar extends AppCompatActivity {

    ArrayList<Color> coloresList;
    ArrayList<String> Lista_Fabricantes; //Nuevo
    Button IdComparar;
    TextView IdRojo;
    TextView IdVerde;
    TextView IdAzul;
    TextView IdReferencia;
    TextView IdFabricante;
    TextView IdColor;
    Spinner IdSpinner; //nuevo
    int g,b, Drojo,Dazul,Dverde,posicionColor,
operacionColor;
    int x = 10000;
    int s = 0; //nuevo



    ConexionSQLiteHelper conn;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_comparar);

```

```
45          conn = new ConexionSQLiteHelper(
   getApplicationContext(),"bd_colores",null,1);
46
47
48
49          IdColor = (TextView) findViewById(R.id.Idco);
50          IdRojo = (TextView) findViewById(R.id.Idro);
51          IdVerde = (TextView) findViewById(R.id.Idve);
52          IdAzul = (TextView) findViewById(R.id.Idaz);
53          IdReferencia = (TextView) findViewById(R.id.Idrefe
   );
54          IdFabricante = (TextView) findViewById(R.id.Idfa);
55          IdComparar = (Button) findViewById(R.id.Idcomp);
56          IdSpinner = (Spinner) findViewById(R.id.Idspinner)
   ; //nuevo
57
58
59
60
61        consultarListaColores();
62          ArrayAdapter<CharSequence> adapter = new
   ArrayAdapter(this, android.R.layout.simple_spinner_item,
   Lista_Fabricantes); //Nuevo
63
64          IdSpinner.setAdapter(adapter);
65
66          Intent miIntent = getIntent();
67          Bundle extras = miIntent.getExtras();
68
69          String dato_color = extras.getString("COLOR");
70          IdColor.setText(dato_color);
71          for (int i = 0; i < dato_color.length(); i++){
72              char x=dato_color.charAt(i);
73              if(x == 'V') { g = i; }
74              if(x == 'A') { b = i; }
75          }
76          String rojo = dato_color.substring(1,g);
77          String verde = dato_color.substring(g+1,b);
78          String azul = dato_color.substring(b+1,dato_color.
   length());
79
80          IdRojo.setText("VALOR ROJO: " + rojo);
81          IdAzul.setText("VAOR AZUL: " + azul);
82          IdVerde.setText("VALOR VERDE: " + verde);
83
```

```
 84
 85         //Guardo en estas variables los valores medidos
    como enteros.
 86         Drojo = Integer.parseInt(rojo);
 87         Dverde = Integer.parseInt(verde);
 88         Dazul = Integer.parseInt(azul);
 89
 90
 91
 92         IdComparar.setOnClickListener(new View.
    OnClickListener(){
 93             @Override
 94             public void onClick(View v)
 95             {
 96
 97
 98                 compararvalores();
 99
100
101                 IdReferencia.setText("REFERENCIA: " +
    coloresList.get(posicionColor).getReferencia());// +
    coloresList.get(0).getReferencia().toString());
102                 IdFabricante.setText("FABRICANTE: " +
    coloresList.get(posicionColor).getFabricante());// +
    coloresList.get(0).getFabricante().toString());
103
104                 x = 10000;
105
106             }
107         });
108
109
110     }
111
112
113     private void compararvalores() {
114
115         for(int i = 0;i<coloresList.size();i++){
116             String uperf = coloresList.get(i).
    getFabricante().toUpperCase();
117             if(uperf.equals(IdSpinner.getSelectedItem())
    || IdSpinner.getSelectedItem().equals("Todos") ){
118
119             int Rguardado = coloresList.get(i).getRojo();
120             int Vguardado = coloresList.get(i).getVerde()
```

```
120 ;
121             int Aguardado = coloresList.get(i).getAzul();
122
123             operacionColor =(int) Math.sqrt((Dazul-
    Aguardado)*(Dazul-Aguardado)+(Dverde-Vguardado)*(Dverde-
    Vguardado)+(Drojo-Rguardado)*(Drojo-Rguardado));
124             if(operacionColor < x) {
125                 x = operacionColor;
126                 posicionColor = i;
127             }
128
129             }
130         }
131
132     }
133
134
135
136
137
138
139     private void consultarListaColores() {
140
141         SQLiteDatabase db=conn.getReadableDatabase();
142         Color color = null;
143         coloresList =new ArrayList<Color>();
144
145         Cursor cu = db.rawQuery("SELECT * FROM " +
    Utilidades.TABLA_COLORES ,null);
146
147         while(cu.moveToNext()){
148             color = new Color();
149             color.setRojo(cu.getInt(0));
150             color.setVerde(cu.getInt(1));
151             color.setAzul(cu.getInt(2));
152             color.setReferencia(cu.getString(3));
153            color.setFabricante(cu.getString(4));
154
155             coloresList.add(color);
156        }
157         obtenerFabricantes(); //nuevo
158     }
159
160
161
```

```
162     private void obtenerFabricantes() {
163         Lista_Fabricantes= new ArrayList<String>();
164         Lista_Fabricantes.add("Seleccione Fabricante");
165         Lista_Fabricantes.add("Todos");
166         String fabri;
167         String fab;
168         for(int q=10;q<coloresList.size();q++){
169             fabri = coloresList.get(q).getFabricante();
170             String uperfabri = fabri.toUpperCase();
171             for(int w=0;w<Lista_Fabricantes.size();w++ )
172             {
173
174                 fab = Lista_Fabricantes.get(w);
175                 String uperfab = fab.toUpperCase();
176                 if(uperfabri.equals(uperfab))
177                 {
178
179                     s++;
180                 }
181
182             }
183
184             if(s == 0) {
185                 Lista_Fabricantes.add(coloresList.get(q).
    getFabricante().toString().toUpperCase());
186
187             }
188             s = 0;
189         }
190
191
192     }
193
194
195
196
197 }
198
199
200
201
```

```java
package com.estudiantes.controbt;

import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.bluetooth.BluetoothSocket;
import android.content.Intent;
import android.os.Handler;
import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;
import android.support.v7.widget.ButtonBarLayout;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;

import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.util.UUID;

public class UserInterfaz extends AppCompatActivity {

    //1)
    Button IdEncender,IdDesconectar,IdRegistro,
IdComprobar;
    TextView IdBufferIn;


    //----------------------------------------
    Handler bluetoothIn;
    final int handlerState = 0;
    private BluetoothAdapter btAdapter = null;
    private BluetoothSocket btSocket = null;
    private StringBuilder DataStringIN = new StringBuilder
();
    private ConnectedThread MyConexionBT;
    // Identificador unico de servicio - SPP UUID
    private static final UUID BTMODULEUUID = UUID.
fromString("00001101-0000-1000-8000-00805F9B34FB");
    // String para la direccion MAC
    private static String address = null;
    //----------------------------------------

    @Override
```

```
43      protected void onCreate(Bundle savedInstanceState) {
44          super.onCreate(savedInstanceState);
45          setContentView(R.layout.activity_user_interfaz);
46
47          ConexionSQLiteHelper conn=new ConexionSQLiteHelper
    (this,"bd_colores",null,1);
48
49          IdEncender = (Button) findViewById(R.id.IdEncender
    );
50          IdDesconectar = (Button) findViewById(R.id.
    IdDesconectar);
51          IdBufferIn = (TextView) findViewById(R.id.
    IdBufferIn);
52          IdRegistro = (Button) findViewById(R.id.Idregistro
    );
53          IdComprobar = (Button) findViewById(R.id.
    Idcomprobar);
54
55
56          bluetoothIn = new Handler() {
57              public void handleMessage(android.os.Message
    msg) {
58                  if (msg.what == handlerState) {
59                      String readMessage = (String) msg.obj;
60                      DataStringIN.append(readMessage);
61
62                      int endOfLineIndex = DataStringIN.
    indexOf("#");
63
64                      if (endOfLineIndex > 0) {
65                          String dataInPrint = DataStringIN.
    substring(0, endOfLineIndex);
66                          IdBufferIn.setText(dataInPrint);
67                          DataStringIN.delete(0,
    DataStringIN.length());
68                      }
69                  }
70              }
71          };
72
73          btAdapter = BluetoothAdapter.getDefaultAdapter();
74          VerificarEstadoBT();
75
76
77          IdRegistro.setOnClickListener(new View.
```

```
 77 OnClickListener(){
 78             @Override
 79             public void onClick(View v)
 80             {
 81                 //Al pulsar, envio lo que haya en Dato y
    cambio a la siguiente actividad.
 82                 String color =  IdBufferIn.getText().
    toString();
 83                 Intent miIntent = new Intent(UserInterfaz
    .this,prueba.class);
 84                 miIntent.putExtra("COLOR", color);
 85                 startActivity(miIntent);
 86             }
 87                                     });
 88
 89         IdComprobar.setOnClickListener(new View.
    OnClickListener(){
 90             @Override
 91             public void onClick(View v)
 92             {
 93                 //Al pulsar, envio lo que haya en Dato y
    cambio a la siguiente actividad.
 94                 String color =  IdBufferIn.getText().
    toString();
 95                 Intent miIntent = new Intent(UserInterfaz
    .this,Comparar.class);
 96                 miIntent.putExtra("COLOR", color);
 97                 startActivity(miIntent);
 98             }
 99         });
100
101
102         IdEncender.setOnClickListener(new View.
    OnClickListener() {
103             public void onClick(View v)
104             {
105                 MyConexionBT.write("1");
106             }
107         });
108
109
110
111         IdDesconectar.setOnClickListener(new View.
    OnClickListener() {
112             public void onClick(View v) {
```

```
113                 if (btSocket!=null)
114                 {
115                     try {btSocket.close();}
116                     catch (IOException e)
117                     { Toast.makeText(getBaseContext(), "
    Error", Toast.LENGTH_SHORT).show();;}
118                 }
119                 finish();
120             }
121         });
122     }
123 
124     private BluetoothSocket createBluetoothSocket(
    BluetoothDevice device) throws IOException
125     {
126         //crea un conexion de salida segura para el
    dispositivo
127         //usando el servicio UUID
128         return device.createRfcommSocketToServiceRecord(
    BTMODULEUUID);
129     }
130 
131     @Override
132     public void onResume()
133     {
134         super.onResume();
135         //Consigue la direccion MAC desde
    DeviceListActivity via intent
136         Intent intent = getIntent();
137         //Consigue la direccion MAC desde
    DeviceListActivity via EXTRA
138         address = intent.getStringExtra(DispositivosBT.
    EXTRA_DEVICE_ADDRESS);
139         //Setea la direccion MAC
140         BluetoothDevice device = btAdapter.
    getRemoteDevice(address);
141 
142         try
143         {
144             btSocket = createBluetoothSocket(device);
145         } catch (IOException e) {
146             Toast.makeText(getBaseContext(), "La
    creacción del Socket fallo", Toast.LENGTH_LONG).show();
147         }
148         // Establece la conexión con el socket Bluetooth.
```

```
149         try
150         {
151             btSocket.connect();
152         } catch (IOException e) {
153             try {
154                 btSocket.close();
155             } catch (IOException e2) {}
156         }
157         MyConexionBT = new ConnectedThread(btSocket);
158         MyConexionBT.start();
159     }
160 
161     @Override
162     public void onPause()
163     {
164         super.onPause();
165         try
166         { // Cuando se sale de la aplicación esta parte
    permite
167             // que no se deje abierto el socket
168             btSocket.close();
169         } catch (IOException e2) {}
170     }
171 
172     //Comprueba que el dispositivo Bluetooth Bluetooth
    está disponible y solicita que se active si está
    desactivado
173     private void VerificarEstadoBT() {
174 
175         if(btAdapter==null) {
176             Toast.makeText(getBaseContext(), "El
    dispositivo no soporta bluetooth", Toast.LENGTH_LONG).
    show();
177         } else {
178             if (btAdapter.isEnabled()) {
179             } else {
180                 Intent enableBtIntent = new Intent(
    BluetoothAdapter.ACTION_REQUEST_ENABLE);
181                 startActivityForResult(enableBtIntent, 1)
    ;
182             }
183         }
184     }
185 
186     //Crea la clase que permite crear el evento de
```

```
186 conexion
187     private class ConnectedThread extends Thread
188     {
189         private final InputStream mmInStream;
190         private final OutputStream mmOutStream;
191 
192         public ConnectedThread(BluetoothSocket socket)
193         {
194             InputStream tmpIn = null;
195             OutputStream tmpOut = null;
196             try
197             {
198                 tmpIn = socket.getInputStream();
199                 tmpOut = socket.getOutputStream();
200             } catch (IOException e) { }
201             mmInStream = tmpIn;
202             mmOutStream = tmpOut;
203         }
204 
205         public void run()
206         {
207             byte[] buffer = new byte[256];
208             int bytes;
209 
210             // Se mantiene en modo escucha para
    determinar el ingreso de datos
211             while (true) {
212                 try {
213                     bytes = mmInStream.read(buffer);
214                     String readMessage = new String(
    buffer, 0, bytes);
215                     // Envia los datos obtenidos hacia el
     evento via handler
216                     bluetoothIn.obtainMessage(
    handlerState, bytes, -1, readMessage).sendToTarget();
217                 } catch (IOException e) {
218                     break;
219                 }
220             }
221         }
222         //Envio de trama
223         public void write(String input)
224         {
225             try {
226                 mmOutStream.write(input.getBytes());
```

```
227             }
228             catch (IOException e)
229             {
230                 //si no es posible enviar datos se cierra
     la conexión
231                 Toast.makeText(getBaseContext(), "La
    Conexión fallo", Toast.LENGTH_LONG).show();
232                 finish();
233             }
234         }
235     }
236 }
237
```

```java
package com.estudiantes.controbt;

import android.support.v7.app.AppCompatActivity;
import android.os.Bundle;

import android.bluetooth.BluetoothAdapter;
import android.bluetooth.BluetoothDevice;
import android.content.Intent;
import android.util.Log;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import android.widget.TextView;
import android.widget.Toast;

import java.util.Set;

public class DispositivosBT extends AppCompatActivity {

    //1)
    // Depuración de LOGCAT
    private static final String TAG = "DispositivosBT";
    // Declaracion de ListView
    ListView IdLista;
    // String que se enviara a la actividad principal, mainactivity
    public static String EXTRA_DEVICE_ADDRESS = "device_address";

    // Declaracion de campos
    private BluetoothAdapter mBtAdapter;
    private ArrayAdapter mPairedDevicesArrayAdapter;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_dispositivos_bt);
    }

    @Override
    public void onResume()
    {
        super.onResume();
        //----------------------------------
```

```
44          VerificarEstadoBT();
45
46          // Inicializa la array que contendra la lista de
    los dispositivos bluetooth vinculados
47          mPairedDevicesArrayAdapter = new ArrayAdapter(this
    , R.layout.nombre_dispositivos);
48          // Presenta los disposisitivos vinculados en el
    ListView
49          IdLista = (ListView) findViewById(R.id.IdLista);
50          IdLista.setAdapter(mPairedDevicesArrayAdapter);
51          IdLista.setOnItemClickListener(
    mDeviceClickListener);
52          // Obtiene el adaptador local Bluetooth adapter
53          mBtAdapter = BluetoothAdapter.getDefaultAdapter();
54          // Obtiene un conjunto de dispositivos actualmente
     emparejados y agregua a 'pairedDevices'
55          Set <BluetoothDevice> pairedDevices = mBtAdapter.
    getBondedDevices();
56          // Adiciona un dispositivos previo emparejado al
    array
57          if (pairedDevices.size() > 0)
58          {
59              for (BluetoothDevice device : pairedDevices) {
60                  mPairedDevicesArrayAdapter.add(device.
    getName() + "\n" + device.getAddress());
61              }
62          }
63      }
64
65      // Configura un (on-click) para la lista
66      private AdapterView.OnItemClickListener
    mDeviceClickListener = new AdapterView.OnItemClickListener
    () {
67          public void onItemClick(AdapterView av, View v,
    int arg2, long arg3) {
68
69              // Obtener la dirección MAC del dispositivo,
    que son los últimos 17 caracteres en la vista
70              String info = ((TextView) v).getText().
    toString();
71              String address = info.substring(info.length()
    - 17);
72
73              // Realiza un intent para iniciar la siguiente
     actividad
```

```
74              // mientras toma un EXTRA_DEVICE_ADDRESS que
   es la dirección MAC.
75              Intent i = new Intent(DispositivosBT.this,
   UserInterfaz.class);
76              i.putExtra(EXTRA_DEVICE_ADDRESS, address);
77              startActivity(i);
78          }
79      };
80
81      private void VerificarEstadoBT() {
82          // Comprueba que el dispositivo tiene Bluetooth y
    que está encendido.
83          mBtAdapter= BluetoothAdapter.getDefaultAdapter();
84          if(mBtAdapter==null) {
85              Toast.makeText(getBaseContext(), "El
   dispositivo no soporta Bluetooth", Toast.LENGTH_SHORT).
   show();
86          } else {
87              if (mBtAdapter.isEnabled()) {
88                  Log.d(TAG, "...Bluetooth Activado...");
89              } else {
90                  //Solicita al usuario que active
   Bluetooth
91                  Intent enableBtIntent = new Intent(
   BluetoothAdapter.ACTION_REQUEST_ENABLE);
92                  startActivityForResult(enableBtIntent, 1)
   ;
93
94              }
95          }
96      }
97  }
```

```java
package com.estudiantes.controbt;

import android.content.Context;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;

import com.estudiantes.controbt.utilidades.Utilidades;

public class ConexionSQLiteHelper extends SQLiteOpenHelper {



    public ConexionSQLiteHelper(Context context, String name, SQLiteDatabase.CursorFactory factory, int version) {
        super(context, name, factory, version);
    }

    @Override
    public void onCreate(SQLiteDatabase db) {
        db.execSQL(Utilidades.CREAR_TABBLA_COLOR);

    }

    @Override
    public void onUpgrade(SQLiteDatabase db, int version_antigua, int version_nueva) {
        db.execSQL("DROP TABLE IF EXISTS colores");
        onCreate(db);
    }
}

```

```java
package com.estudiantes.controbt.entidades;

public class Color {

    private Integer Rojo;
    private Integer Verde;
    private Integer Azul;
    private String Fabricante;
    private String Referencia;

    public Color(Integer rojo, Integer verde, Integer azul
, String fabricante, String referencia) {
        this.Rojo = rojo;
        this.Verde = verde;
        this.Azul = azul;
        this.Fabricante = fabricante;
        this.Referencia = referencia;
    }

    public Color() {
    }

    public void setRojo(Integer rojo) {
        Rojo = rojo;
    }

    public void setVerde(Integer verde) {
        Verde = verde;
    }

    public void setAzul(Integer azul) {
        Azul = azul;
    }

    public void setFabricante(String fabricante) {
        Fabricante = fabricante;
    }

    public void setReferencia(String referencia) {
        Referencia = referencia;
    }

    public Integer getRojo() {
        return Rojo;
    }
```

```
45
46      public Integer getVerde() {
47          return Verde;
48      }
49
50      public Integer getAzul() {
51          return Azul;
52      }
53
54      public String getFabricante() {
55          return Fabricante;
56      }
57
58      public String getReferencia() {
59          return Referencia;
60      }
61  }
62
```

```java
package com.estudiantes.controbt.utilidades;

public class Utilidades {

    //Constantes campos

    public static final String TABLA_COLORES="color";
    public static final String CAMPO_ROJO="rojo";
    public static final String CAMPO_VERDE="verde";
    public static final String CAMPO_AZUL="azul";
    public static final String CAMPO_REFERENCIA="referencia";
    public static final String CAMPO_FABRICANTE="fabricante";


   public static final String CREAR_TABBLA_COLOR = "CREATE TABLE "+TABLA_COLORES+" ("+CAMPO_ROJO+" INTEGER, "+CAMPO_VERDE+" INTEGER, "+CAMPO_AZUL+" INTEGER, "+CAMPO_REFERENCIA+" TEXT, "+CAMPO_FABRICANTE+" TEXT)";



}

```